Tampereen yliopisto

Sami Joutsijoki

KVANTTITIETOKONEIDEN VAIKUTUS KRYPTOGRAFIAAN

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Toukokuu 2019

TIIVISTELMÄ

Sami Joutsijoki: Kvanttitietokoneiden vaikutus kryptografiaan
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2019

Tässä tutkielmassa selvitetään kvanttitietokoneiden vaikutusta moderniin kryptografiaan. Kryptografialla tarkoitetaan salakirjoituksen tutkimusta, missä tarkoituksena on kahden osapuolen välisen viestinnän salaaminen siten, että kolmannen osapuolen on mahdotonta lukea viestejä tietämättä käytettyä salausmenetelmää tai moderneissa menetelmissä tietämättä salaukseen käytettyä salausavainta. Kvanttitietokoneilla tarkoitetaan tietokoneita tai kvanttiprosessointiyksiköitä, jotka hyödyntävät kvanttimekaniikan lainalaisuuksia laskutoimituksissaan. Tämä mahdollistaa tietyn tyyppisten ongelmien ratkaisemisen jopa eksponentiaalisesti nopeammassa ajassa verrattuna klassiseen tietokoneeseen. Lähes kaikki nykyhetken toiminnassa olevat julkisen avaimen kryptosysteemit perustuvat sellaisiin matemaattisiin ongelmiin, joiden ratkaiseminen kvanttitietokoneella on teoriassa mahdollista järkevässä ajassa. Vaikka kvanttitietokoneet eivät ole vielä siinä pisteessä, että ne pystyisivät murtamaan nykyhetken salauksia, on niiden kehitys ottanut valtavasti tuulta alleen. Kvanttitietokoneiden kehitys on myös suunnannut tutkijoiden mielenkiinnon vaihtoehtoisiin ”kvanttiresistantteihin” kryptosysteemeihin. Nämä systeemit perustuvat sellaisiin ongelmiin, joihin nykypäivän kvanttialgoritmit eivät tarjoa klassisia algoritmeja nopeampia ratkaisuja. Tutkielmassa selvitetään myös, millaisia alttiina olevia kryptosysteemeitä hyödyntäviä arkipäivän sovelluksia ohjelmistotalo Solitalla on käytössä, pohditaan tällaisen kvanttikonehyökkäyksen todennäköisyyttä ja etsitään korvaavia kvanttiresistantteja kryptosysteemeitä.

Tutkielma jakautuu neljään päälukuun. Luvussa 2 kerrotaan hieman kryptografian historiasta ja esitellään nykyajan käytetyimpiä kryptografisia primitiivejä. Luvussa 3 avataan kvanttitietokoneiden arkkitehtuuria ja verrataan niitä klassisiin tietokoneisiin sekä esitellään tunnetuimmat kvanttialgoritmit ja selvitetään kvanttitietokoneiden nykytilaa. Luvussa 4 esitellään lyhyesti ne matemaattiset ongelmat, joihin kvanttiresistantit kryptosysteemit perustuvat sekä mainitaan muutama tällainen kryptosysteemi nimeltä. Lopuksi luvussa 5 selvitetään Solitalla käytössä olevien sovellusten käyttämät kryptosysteemit ja etsitään korvaavia systeemeitä.

Avainsanat: kryptografia, kvanttitietokone, kvanttiresistantti kryptografia

Sisällys

1. Johdanto

Kryptografia tulee kreikan sanoista *kryptós* (salainen) ja *graphein* (kirjoitus). Kryptografia on siis salakirjoituksen tutkimusta, missä tarkoituksena on kahden osapuolen välisen viestinnän salaaminen siten, että kolmannen osapuolen on mahdotonta lukea viestejä tietämättä käytettyä salausmenetelmää, tai moderneissa menetelmissä tietämättä salaukseen käytettyä salausavainta.

Kvanttitietokoneet ovat tietojenkäsittelytieteen eräs suurimmista hiljattaisista saavutuksista. Mooren lakina tunnetun lauseen mukaan tietokoneiden suorituskyky tuplaantuu kahden vuoden välein. Tämä sääntö on pysynyt hämmästyttävän tarkkana 60-luvulta saakka, mutta yleisesti ajatellaan, että tämä kehitys on pysähtymässä. Syynä on se, että sähköisten laitteiden pienentyessä kvanttimekaniikan ilmiöt alkavat aiheuttamaan liikaa häiriöitä. Ratkaisuksi tähän ongelmaan onkin ajateltu kvanttitietokoneita, sillä niiden arkkitehtuuri perustuu kvanttimekaniikan ilmiöihin. Kvanttitietokoneilla kehitetyt algoritmit tarjoavat tiettyjen ongelmien ratkaisemiseksi jopa eksponentiaalisia nopeusparannuksia suhteessa klassisilla tietokoneilla kehitettyihin parhaisiin mahdollisiin algoritmeihin. Tämä johtaa muun muassa siihen, että nykyään käytetyimmät julkisen avaimen kryptosysteemit muuttuvat mahdollisiksi murtaa realistisessa ajassa. [Nielsen & Chuang 2010]

Tässä tutkielmassa selvitetään kvanttitietokoneiden vaikutusta moderniin kryptografiaan sekä tehdään selvitys ohjelmistotalo Solitan salausprotokollia käyttävien ohjelmistojen tilasta ja tarpeen mukaan ehdotetaan muutostoimenpiteitä. Tutkielma on jaettu neljään päälukuun. Luvussa 2 kerrotaan tarkemmin kryptografian historiasta sekä käytetyimmistä moderneista yksityisen ja julkisen avaimen salausmenetelmistä. Luvussa 3 avataan kvanttitietokoneiden arkkitehtuuria, vertaillaan niitä klassisiin tietokoneisiin sekä esitetään muutama kvanttialgoritmi, joilla käytännössä murretaan moderni kryptografia. Luvussa 3 myös selvitetään, mikä on kvanttitietokoneiden nykytila ja pohditaan, milloin kvanttitietokoneet alkavat olla arkipäivää. Luvussa 4 kuvataan lyhyesti muutamia vaihtoehtoisia kryptosysteemejä, joiden murtamista varten ei vielä ole keksitty kvanttialgoritmeja, ja jotka ovat täten lupaavia kandidaatteja tulevaisuuden kryptosysteemeiksi. Lopulta luvussa 5 kerrotaan Solitan käyttämien keskeisten sovellusten salausmenetelmien tilasta ja pohditaan, mitä toimenpiteitä voidaan tehdä kvanttitietokoneita varten.

2. Kryptografia

Kryptografian tutkimuksen tavoitteena on kehittää menetelmiä, joilla kahden osapuolen (Alice ja Bob) välinen kommunikaatio pysyy turvattuna, vaikka heidän väliset viestinsä kaapattaisiin matkalla kolmannen osapuolen toimesta. Ongelmina on siis sekä viestien muuttaminen sellaisiksi, että ne eivät ole luettavissa ilman lisätietoa käytetystä salausmenetelmästä, sekä moderneissa kryptosysteemeissä se, että viestien lähettäjä on jollain tavalla varmennettavissa. Viestien salaus on ollut varsin relevantti ongelma jo pitkän aikaa, sillä ensimmäiset salausmenetelmät ovat olleet olemassa jo antiikin ajoista lähtien.

2.1. Aika ennen tietokoneita

Eräs tunnetuimmista antiikin ajan salausmenetelmistä on Caesarin salausmenetelmä (Caesar's cipher), joka on nimetty tunnetuimman käyttäjänsä Julius Caesarin mukaan. Tässä menetelmässä jokainen selkokielisen tekstin (plaintext) kirjain siirretään aakkostossa kolmen kirjaimen verran eteenpäin saadaksemme salatun tekstin (ciphertext). Vastaavasti salattu teksti puretaan siirtämällä jokainen kirjain kolmen kirjaimen verran taaksepäin. Caesarin aikaan tämä menetelmä oli ilmeisesti tarpeeksi turvallinen kirjeiden vaihtoon, mutta nykypäivänä tietokoneiden avustuksella tämän menetelmän pystyy murtamaan helposti, vaikka emme tietäisikään kirjainten siirtoarvoa (shift value). Aakkostossa on äärellinen ja hyvin vähäinen määrä alkioita, jolloin voimme yksinkertaisesti testata jokaista arvoa ja tutkia onko tulos järkevä. [Aumasson, 2018]

Caesarin menetelmä oli varsin pitkään paras tunnettu salausmenetelmä, kunnes noin 1500 vuotta myöhemmin italialainen Giovan Battista Bellaso paransi menetelmää ottamalla käyttöön salausavaimen. Menetelmän nimi on Vigenéren salaus (Vigenére cipher), joka tulee ranskalaisen Blaise de Vigenéren nimestä. Hän taas kehitti eri menetelmän, mutta historian virheen ansiosta Vigenéren nimi pysyi. Tässä menetelmässä selkokielen tekstin kirjaimet siirretään salausavaimen kirjainten perusteella, missä salausavaimen sijainti aakkostossa määrittelee siirtoarvon. Avain on yleensä lyhyempi kuin salattava teksti ja avainta toistetaan, kunnes avain on tekstin pituinen. Esimerkiksi jos salattava teksti on ”Huomenna on poutaa” ja salausavain ”ABC” määräytyisi salateksti taulukon 1 mukaan.

Selkokielinen teksti	H	U	O	M	E	N	N	A	O	N	P	O	U	T	A	A
Salausavain toistettuna	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A
Siirtoarvo	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1
Salattu teksti	I	W	R	N	G	Q	O	C	R	O	R	R	V	V	D	B

Taulukko 1. Tekstin salaus Vigenéren menetelmällä.

Vigenéren menetelmä on parannus Caesarin menetelmään nähden, sillä nyt kolmannen osapuolen pitäisi tietää salausavain pystyäkseen murtamaan salauksen. Tässäkin menetelmässä on kuitenkin heikkoutensa. Huomataan, että tekstin seitsemäs ja kymmenes kirjain ovat molemmat N ja ne salautuvat samaksi kirjaimeksi O. Sama tapahtuu yhdeksännen ja kahdennentoista kirjaimen O kanssa, molemmat salautuvat kirjaimeksi R. Yhteistä näillä kohdilla on etäisyys toistensa kanssa, 10-7 = 3 ja 12-9 = 3. Tämä antaa viitteitä siitä, että salausavaimen koko olisi 3, sillä sama selkokielisen tekstin kirjain voi salautua samaksi salatun tekstin kirjaimeksi vain, jos se on salattu samalla kirjaimella. Näin ollen voidaan päätellä, että salausavaimen kirjaimet toistuvat joka kolmannen kirjaimen kohdalla. Tästä eteenpäin voidaan käyttää frekvenssianalyysia varsinaisen salausavaimen murtamiseen, missä lasketaan kirjainten esiintymistiheyksiä ja verrataan niitä luonnollisen kielen aakkosten esiintymistiheyteen. [Aumasson, 2018]

Edellä mainittujen heikkouksien takia modernit tietokoneilla toteutettavat salausmenetelmät eivät perustu kirjainkohtaiseen salaamiseen, vaan ne tulkitsevat salattavan tekstin tavuina ja käyttävät erilaisia operaatioita, joilla tavut sekoitetaan mahdollisimman sattumanvaraisesti salausavaimen perusteella. Seuraavassa alaluvussa tutustutaan hieman maailman käytetyimpään, ”de facto” salausmenetelmään nimeltä AES.

2.2. Advanced Encryption Standard

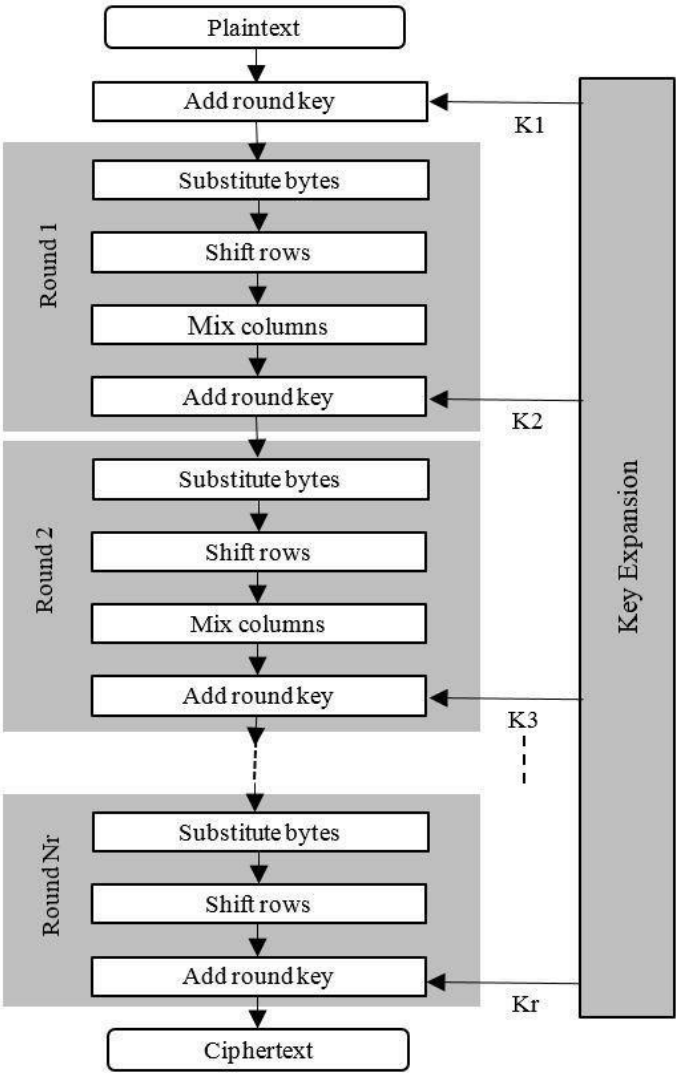
Yhdysvaltalainen standardisointiyhtiö National Institute of Standards and Technology (NIST) järjesti vuosina 1997-2000 kilpailun, jossa tavoitteena oli korvata vanhentunut salausstandardi DES (Data Encryption Standard) joka oli hidas ja tarjosi salausavaimen kokoon nähden huonon turvallisuustason. Kilpailuun osallistuneista 15:a kandidaatista voittajaksi valittiin salausmenetelmä Rijndael, joka on nimetty kehittäjiensä Joan Daemenin ja Vincent Rijmenin mukaan. NIST nimesi menetelmän Advanced Encryption Standardiksi (AES), joka on nykyään maailman käytetyin symmetrisen avaimen salausmenetelmä [Daemen & Rijmen 1998]. Symmetrisen avaimen menetelmä on salausmenetelmä, missä sekä salauksessa että salauksen purkamisessa käytettävä avain on identtinen. Edellä mainittu Vigenéren menetelmä oli myös symmetrinen. [Aumasson, 2018].

AES käsittelee 128 bitin kokoisia lohkoja kerrallaan. Salausavaimen koko on joko 128, 192 tai 256 bittiä. Lohkoja käsitellään kaksiulotteisina 4x4 matriiseina, missä jokainen alkio on yksi tavu (8 bittiä).

s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

Taulukko 2. AES:n sisäinen tila, missä s_n on 1 tavu.

Salaus toimii niin, että riippuen salausavaimen koosta toistetaan kuvan 1 mukaisia askeleita joko 10 (128-bittisellä avaimella), 12 (192-bittisellä avaimella) tai 14 (256-bittisellä avaimella) kertaa.



Kuva 1. AES-salauksen vaiheet [Aumasson, 2018].

Salauksen eri vaiheet koostuvat osista ”Key Expansion”, ”Add round key”, ”Substitute bytes”, ”Shift rows” ja ”Mix columns”. ”Key Expansion” luo salausavaimesta uuden avaimen jokaiselle askeleelle, jotta hyökkääjä ei pystyisi päättelemään jokaisen askeleen salausavainta yhden avaimen perusteella. ”Add round key” -askeleessa suoritetaan XOR-operaatio sisäisen tilan ja kyseisen askeleen salausavaimen kesken, jotta salaus riippuisi salausavaimesta. ”Substitute bytes” -operaatio vaihtaa sisäisen tilan jokaisen tavun toiseen tavuun hakutaulukon perusteella, jotta salausmenetelmään tulisi epälineaarisuutta. ”Shift rows” siirtää sisäisen tilan tavuja rivien perusteella, jotta muutokset tietyssä selkotekstin sarakkeessa vaikuttaisivat muihin salatun tekstin sarakkeisiin. ”Mix columns” suorittaa jokaiselle sarakkeelle saman lineaarisen muunnoksen, jotta muutos yhdessä selkotekstin tavussa vaikuttaisi muihin salatun tekstin tavuihin. ”Shift rows” ja ”Mix columns” -askeleet tuovat siis *diffuusia* salausmenetelmään, joka tarkoittaa sitä, että yhdenkin bitin muutos selkokielisessä tekstissä pitäisi vaikuttaa jokaiseen salatun tekstin bittiin. [Aumasson, 2018]

AES on maailman käytetyin salausmenetelmä, mutta koska se on *symmetrinen salausmenetelmä*, on sillä eräs ongelma. Kommunikaation molempien osapuolien on tiedettävä salausavain etukäteen, jotta salattu kommunikaatio olisi mahdollista. Miten siis välittää salausavain turvallisesti internetin välityksellä sillä oletuksella, että kaikki tiedonsiirto on mahdollista kaapata ja lukea kolmannen osapuolen toimesta? Tähän ongelmaan keksittiin 1970-luvulla nerokas ratkaisu, metodi nimeltään RSA.

2.3. RSA

RSA on ensimmäinen *asymmetrinen salausmenetelmä*, jossa kommunikaation osapuolilla on sekä yksityinen, että julkinen avain. RSA on nimetty kehittäjiensä Ron Rivestin, Adi Shamirin ja Leonard Adlemanin mukaan. RSA-menetelmä julkaistiin vuonna 1978 ja se perustui muun muassa muutamaa vuotta aiemmin julkaistuun Whitfield Diffien ja Martin Hellmanin ideaan julkisen avaimen kryptografiasta. Diffie ja Hellman julkaisivat ideansa vuonna 1976, mutta heillä ei tuolloin ollut varsinaista implementaatiota. [Rivest et al. 1978]

Julkisen avaimen kryptosysteemit toimivat niin, että osapuolella A on yksityinen avain a_{priv} sekä julkinen avain a_{pub} ja osapuolella B on vastaavasti avaimet b_{priv} ja b_{pub}. Osapuoli A generoi yksityisen avaimensa a_{priv} pohjalta julkisen avaimen

$$a_{pub} = \boldsymbol{f}(a_{priv})$$

ja lähettää tämän osapuolelle B. Osapuolen B saatua A:n julkisen avaimen a_{pub} yhdistää hän tämän oman yksityisen avaimensa b_{priv} kanssa saaden salaisen arvon

$$c = \boldsymbol{f}(a_{pub}, b_{priv})$$

Vastaavasti B tekee saman toimenpiteen, jolloin A:lla on sama arvo

$$c = \boldsymbol{f}(b_{pub}, a_{priv}) = \boldsymbol{f}(a_{pub}, b_{priv}).$$

Näin osapuolet A ja B ovat muodostaneet jaetun salaisuuden, salausavaimen *c*, jota voidaan käyttää esimerkiksi AES-kryptosysteemin salausavaimena. Julkisen avaimen kryptosysteemien idea on siinä, että julkisten avainten a_{pub} ja b_{pub} perusteella ei pystytä päättelemään yksityisiä avaimia a_{priv} ja b_{priv}, eli toisinsanottuna funktion $\boldsymbol{f}(a_{\text{priv}})$ on oltava sellainen, että $\boldsymbol{f^{-1}}(a_{\text{pub}})$ on laskettavissa ainoastaan niin sanotulla brute force -menetelmällä kokeilemalla jokaista mahdollista a_{pub} arvoa.

Kaikki asymmetriset salausmenetelmät perustuvat johonkin matemaattiseen ongelmaan, missä käytetään *yhden suunnan funktiota*. Yhden suunnan funktio on funktio ***f***, mikä on helppo laskea millä tahansa arvolla, mutta minkä alkukuva on vaikea laskea satunnaisella arvolla, eli

$$y = \boldsymbol{f}(x)$$

on helposti laskettavissa, mutta

$$x = \boldsymbol{f}^{-1}(y)$$

ei. RSA:n taustalla oleva matemaattinen ongelma on *kokonaislukujen tekijöihinjako*, jossa ongelmana on löytää suuren kokonaisluvun $N = pq$ alkulukutekijät p ja q. RSA-kryptosysteemi toimii siten, että osapuoli A ensin valitsee kaksi satunnaista suurta alkulukua p ja q ja laskee näiden pohjalta $N = pq$. Tämän jälkeen lasketaan $\boldsymbol{\varphi}(N)$, jossa $\boldsymbol{\varphi}$ on Eulerin **φ**-funktio (Euler's phi function). Eulerin **φ**-funktio kertoo positiivisten N:ää pienempien N:n kanssa suhteellisten alkulukujen määrän, eli sellaisten lukujen

$$1 \leq x \leq N$$

määrän, joille pätee $\mathbf{syt}(x, N) = 1$. Tässä tapauksessa määrä on $(p-1)(q-1)$. Seuraavaksi generoidaan satunnainen julkinen eksponentti $e < \boldsymbol{\varphi}(N)$ ja sitä vastaava luku d, jolle pätee

$$ed \bmod \boldsymbol{\varphi}(N) = 1.$$

Nyt on muodostettu osapuolen A julkinen avain, joka sisältää tiedon *e* ja *N*, sekä yksityinen avain, joka sisältää tiedon *d*. A lähettää nyt osapuolelle B julkisen avaimensa *e* ja *N*, joita B käyttää salatakseen tekstin *x*, joka on RSA:n kontekstissa selkokielinen teksti tulkittuna jonain kokonaislukuna. Nyt B:llä on viesti

$$y = xe \bmod N,$$

jonka hän lähettää takaisin A:lle. A pystyy nyt purkamaan salauksen ja saamaan tietoonsa selkokielisen tekstin *x* laskemalla

$$yd \bmod N = (xe)d \bmod N = xed \bmod N = x \cdot 1 \bmod N = x.$$

Todellisuudessa RSA ei toimi aivan näin yksinkertaisesti, vaan salattuun tekstiin *y* lisätään satunnaista ”täytettä” (padding), jotta kahden samalla yksityisellä avaimella salatun tekstin y_1 ja y_2 pohjalta ei voitaisi päätellä mitään alkuperäisestä tekstistä x_1 kertomalla ne keskenään [Aumasson, 2018]. RSA:n implementaatio on erittäin vaikeaa ja muun muassa Boneh [1999] selvitti millaisia hyökkäyksiä RSA:ta kohtaan on tehty sen ensimmäisten 20 vuoden aikana. Vaikka mitään koko järjestelmän tuhoavaa hyökkäystä ei ole löydetty, voi RSA:n implementoida väärin monella tapaa. Siksi muun muassa turvallisten parametrien valinta on äärimmäisen tärkeää.

2.4. Diffie-Hellman

Diffie-Hellman -avaimenvaihtoprotokolla on RSA:n tavoin julkisen avaimen protokolla, missä osapuolilla A ja B on sekä julkinen, että yksityinen salausavain. Sen keksivät Stanfordin yliopiston tutkijat Whitfield Diffie ja Martin Hellman vuoden 1976 julkaisussaan ”New directions in cryptography” [Diffie & Hellman 1976]. DH-protokolla perustuu matemaattiseen ongelmaan nimeltä *diskreetin logaritmin ongelma*, missä ongelmana on löytää *a* luvusta $g^a \bmod p$, missä *g* on jokin kokonaisluku ja *p* on DH-protokollan kontekstissa alkuluku, jolle pätee että (*p* – 1) / 2 on myös alkuluku. Tämä tekee protokollasta vaikeammin murrettavan [Aumasson, 2018].

Diffie-Hellman -protokolla toimii siten, että osapuoli A valitsee satunnaisen luvun *a* *äärellisestä ryhmästä kertolaskun suhteen* $\mathbf{Z}_p^*$. Äärellinen ryhmä $\mathbf{Z}_p^*$ on *ryhmä*, jossa on äärellinen määrä alkioita ja jolle pätevät seuraavat ryhmäaksioomat:

1. Operaatio $\circ$ on suljettu joukossa $\mathbf{G}$: $\forall\, a, b \in \mathbf{G}$: $a \circ b \in \mathbf{G}$

2. Operaatio $\circ$ on liitännäinen: $\forall\, a, b, c \in \mathbf{G}: (a \circ b) \circ c = a \circ (b \circ c)$
3. On olemassa neutraalialkio: $\exists e \in \mathbf{G}: \forall a \in \mathbf{G}: e \circ a = a \circ e = a$
4. Jokaisella alkiolla on käänteisalkio: $\forall a \in \mathbf{G}: \exists a^{-1} \in \mathbf{G}: a \circ a^{-1} = a^{-1} \circ a = e$

Ryhmässä $\mathbf{Z}_p^*$ operaatio on modulaarinen kertolasku $a \cdot b \bmod p$ ja ryhmään kuuluvat kaikki kokonaisluvut väliltä [0, *p*-1], missä *p* on alkuluku. Kun luku *a* on valittu, lasketaan

$$x = g^a \bmod p.$$

Luku *a* on osapuolen A yksityinen avain ja luvut *x, g* ja *p* ovat osana julkista avainta, joka lähetetään osapuolelle B. Nyt B vastaavasti valitsee satunnaisen luvun *b* ryhmästä $\mathbf{Z}_p^*$, muodostaa jaetun salaisen arvon laskemalla

$$c = x^b \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$$

ja lähettää osapuolelle A takaisin oman julkisen avaimensa

$$y = g^b \bmod p.$$

Tämän jälkeen A laskee

$$c = y^a \bmod p = (g^b)^a \bmod p = g^{ba} \bmod p.$$

Avaintenvaihdon jälkeen molemmilla osapuolilla on jaettu salaisuus

$$c = g^{ab} \bmod p = g^{ba} \bmod p$$

jonka avulla generoidaan salattuun kommunikaatioon tarvittava avain.

Nykypäivänä Diffie-Hellman -protokolla usein implementoidaan elliptisten käyrien avulla, jolloin kyseessä on elliptisten käyrien Diffie-Hellman -protokolla (ECDH).

2.5. Elliptiset käyrät

Elliptiset käyrät ovat matemaattisia objekteja, käyriä kaksiulotteisessa avaruudessa, joiden yhtälö määrää kaikki kyseisen käyrän pisteet. Kryptografiassa käytetään useasti Weierstrassin muotoa

$$y^2 = x^3 + ax + b$$

olevia käyriä, missä a ja b ovat parametreja, jotka määräävät käyrän muodon. Tämän lisäksi luvut ovat kokonaislukuja ja ne otetaan modulo p, missä p on alkuluku. Kryptografiassa käytettävät elliptiset käyrät eivät siis visuaalisesti näytä lainkaan käyriltä, vaan joukolta pisteiltä. [Aumasson, 2018]

Elliptisten käyrien pisteille on määritelty yhteenlasku, joka riippuu kyseisten pisteiden arvoista. Olkoon $R = P + Q$ elliptisten käyrien pisteiden P ja Q yhteenlaskettu piste. Tällöin on olemassa kolme kaavaa pisteen R koordinaateille x_R ja y_R:

1. Jos $P = -Q$ tai $Q = -P$, on $R = P + (-P) = -Q + Q = O$, missä O tarkoittaa ”pistettä äärettömyydessä” (point at infinity)
2. Jos $P = Q$, saadaan pisteen R koordinaatit $x_R = m^2 - x_P - x_Q$ ja $y_R = m(x_P - x_R) - y_P$, missä $m = \frac{3{x_P}^2 + a}{2y_P}$ ja a on tässä käyrän yhtälön parametri
3. Muussa tapauksessa kaava on sama kuin kohdassa 2, mutta $m = \frac{y_Q - y_P}{x_Q - x_P}$

Elliptisen käyrän pisteiden kertolasku kP, missä k on kokonaisluku, on määritelty laskemalla yhteen piste P itsensä kanssa k-1 kertaa. Tämä on kuitenkin erittäin hidasta, kun k on suuri luku (esim. 256-bittinen). Tätä voidaan nopeuttaa laskemalla pisteet pareittain yhteen. Esimerkiksi haluttaessa piste 16P laskettaisiin seuraavalla tavalla:

$$P_2 = P + P$$
$$P_4 = P_2 + P_2$$
$$P_8 = P_4 + P_4$$
$$P_{16} = P_8 + P_8$$

Näin saavutetaan sama lopputulos neljällä yhteenlaskulla 15 sijasta. [Aumasson, 2018]

Elliptisten käyrien avulla toteutettu Diffie-Hellman -avaimenvaihto perustuu elliptisten käyrien diskreetin logaritmin ongelmaan, missä osana julkista avainta kulkee piste

$$Q = kP \bmod p.$$

Kokonaisluku k on osapuolen A yksityinen avain ja piste Q julkinen. Systeemi perustuu siihen oletukseen, että pisteestä Q salaisen arvon k päättely on vaikea ongelma, eikä ole täten ratkaistavissa klassisella tietokoneella. Elliptiset käyrät ovat korvanneet monet vanhemmat kryptosysteemit, kuten RSA:n ja klassisen Diffie-Hellmanin, sillä elliptiset käyrät eivät vaadi yhtä suuria numeroita saman turvallisuustason tuottamiseksi. Yleisesti

ottaen p:n ollessa n:n bitin kokoinen tarjoaa se $\frac{n}{2}$-bittisen turvallisuustason. Jos halutaan 128-bittinen turvallisuustaso, missä systeemin murtaminen vaatii keskimäärin 2^{128} operaatiota, riittää siihen 256-bittinen alkuluku [Aumasson, 2018]. Muun muassa Washington [2008] mainitsee kirjassaan, että 4096 bitin kokoinen salausavain RSA-kryptosysteemissä antaa saman turvallisuustason kuin 313 bitin kokoinen avain elliptisiä käyriä käyttävässä systeemissä. Samassa kirjassa mainitaan myös, että Dan Boneh ja Neil Daswani vertailivat RSA:n ja elliptisten käyrien avulla generoitujen salausavainten nopeuksia ja havaitsivat, että 512-bittisen avaimen luominen RSA:lla kesti 3.4 minuuttia, kun taas elliptisillä käyrillä 163-bitin avain luotiin 0.597 sekunnissa.

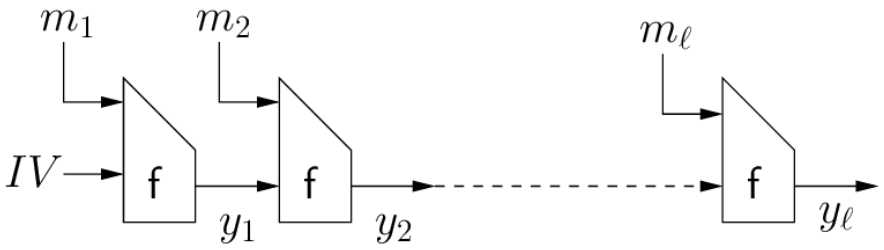
2.6. Kryptografiset hajautusfunktiot

Kryptografiset hajautusfunktiot ovat käytetyimpiä kryptografisia primitiivejä, sillä melkein kaikki kryptografiset protokollat käyttävät ”pinnan alla” kryptografisia hajautusfunktioita. Kryptografiset hajautusfunktiot eroavat tavallisista hajautusfunktioista siten, että tavallisten hajautusfunktioiden ei tarvitse tarjota minkäänlaista kryptografista turvallisuutta [Aumasson, 2018]. Tavallisia hajautusfunktioita käytetään esimerkiksi hajautustaulukko-tietorakenteissa (hash tables), kun taas kryptografisia hajautusfunktioita käytetään muun muassa digitaalisissa allekirjoituksissa, Git-versionhallinnassa ja salasanojen turvaamisessa.

Kryptografinen hajautusfunktio toimii niin, että se ottaa parametrikseen minkä tahansa kokoisen viestin M, ja tuottaa satunnaiselta merkkijonolta näyttävän *tarkisteen* (hash) $H = \mathbf{Hash}(M)$. Tarkiste toimii ikään kuin viestin tunnisteena, sillä yhdenkin bitin muuttuminen viestissä tuottaa täysin erilaisen tarkisteen. Kryptografisen hajautusfunktion pitäisi täyttää kaksi vaatimusta, jotta sitä voisi kutsua turvalliseksi: sen pitäisi olla *alkukuvaresistantti* ja *törmäysresistantti* [Aumasson, 2018]. Alkukuvaresistanssi tarkoittaa sitä, että tiedettäessä jonkin viestin M tarkiste $H = \mathbf{Hash}(M)$, pitäisi olla käytännössä mahdotonta laskea alkukuvaa $M = \mathbf{Hash}^{-1}(H)$. Funktion **Hash** tulisi siis olla yhden suunnan funktio. Toinen vaatimus on törmaysresistanssi, jonka Damgård [1989] määritteli niin, että tiedettäessä jokin viesti M_1 ja sen tarkiste H_1, pitäisi olla käytännössä mahdotonta löytää toinen viesti M_2 jonka tarkiste $H_2 = H_1$. Jos hajautusfunktio ei ole törmäysresistantti, voi se johtaa muun muassa digitaalisten allekirjoitusten väärentämiseen, missä hyökkääjä pystyy generoimaan identtisiä allekirjoituksia kohteena olevan osapuolen viestien ja itse luomiensa viestien välillä tietämättä allekirjoituksessa käytettyä salausavainta [Aumasson, 2018].

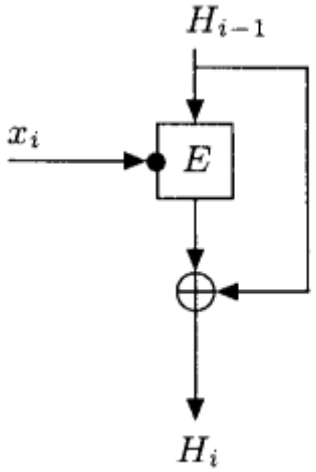
Kaikki yleisimmät kryptografiset hajautusfunktiot, kuten MD4 (Message Digest), MD5, SHA-1 (Secure Hash Algorithm) ja SHA-2 -perheen funktiot on rakennettu Merkle-Damgård -tyylillä, missä viesti M jaetaan keskenään samankokoisiin lohkoihin

$M_1, M_2, \ldots, M_n$, jossa lohkon koko riippuu käytettävästä hajautusfunktiosta (SHA-256 käyttää 512 bitin kokoisia lohkoja ja SHA-512 1024-bittisiä). Tämän jälkeen jokainen lohko lähetetään *ketjutusarvon* mukana vuorotellen *kompressiofunktiolle*, joka muodostaa uuden ketjutusarvon. Viimeisin ketjutusarvo on viestin tarkiste. Tätä havainnollistetaan kuvassa 2.



Kuva 2. Merkle-Damgård -rakenne. IV (initial value) on alussa oleva ketjutusarvo, m_i viestin lohko, *f* kompressiofunktiota ja y_i tietyn hetken ketjutusarvo. Lopullinen ketjutusarvo y_l on viestin tarkiste. [Merkle-Damgård construction, 2019]

Kuvan 2 kompressiofunktio $\boldsymbol{f}$ taas perustuu useimmiten Davies-Meyer -rakenteeseen, jossa kompressiofunktio on rakennettu lohkosalausfunktion pohjalta (kuten esimerkiksi AES-salauksessa), mutta tämän lisäksi *i*-1:n ja *i*:n ketjutusarvon välillä suoritetaan XOR-operaatio, jotta ketjutusarvoa ei pystyttäisi purkamaan lohkosalausfunktiota vastaavalla salauksen purkamisfunktiolla. Kuvassa 3 havainnollistetaan Davies-Meyer -rakennetta.

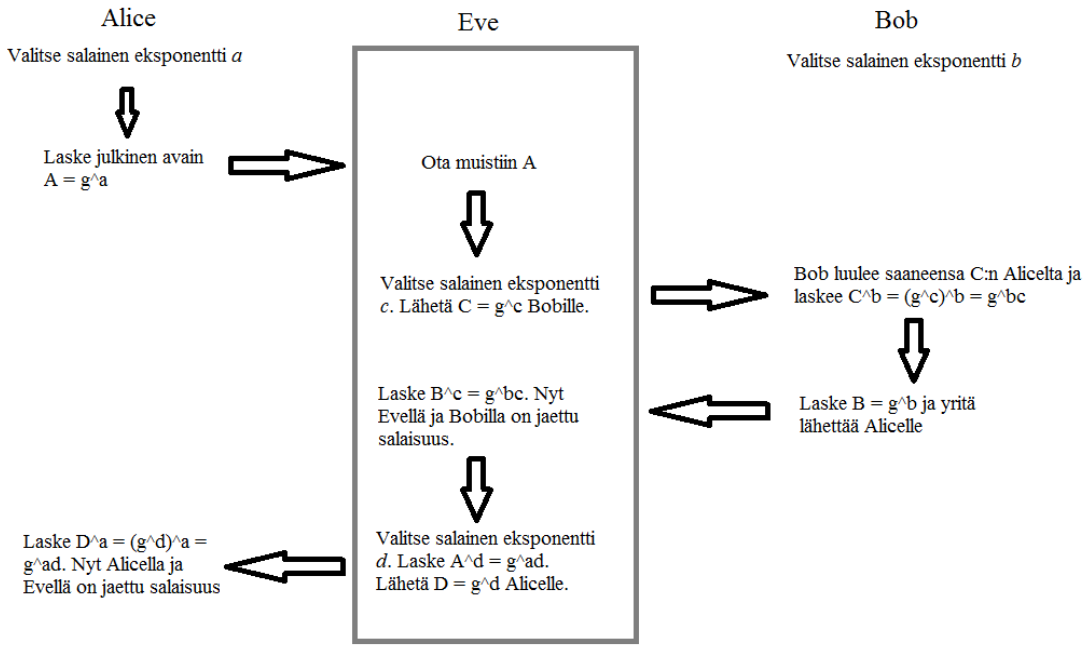


Kuva 3. Davies-Meyer -rakenne kompressiofunktiolle. x_i on viestin lohko, joka toimii ikään kuin salausavaimena salausfunktion **E** näkökulmasta ja H_{i-1} sekä H_i ovat ketjutusarvoja. [Davies-Meyer construction, 2019]

On myös olemassa muita tapoja implementoida kryptografisia hajautusfunktioita, mutta edellä mainitut ovat kaikista yleisimpiä. [Aumasson, 2018]

2.7. Kommunikaation autentikointi

Modernissa maailmassa pelkkä osapuolten välinen viestien salaus ei ole enää riittävä viestinnän turvauksen keino. Tietokoneiden välisessä viestinnässä on mahdollista ujuttautua osapuolten väliin, eli tehdä niin sanottu man in the middle -hyökkäys. Tässä hyökkäyksessä osapuolen A (Alicen) ja B (Bobin) välille ujuttautuu Eve, joka teeskentelee Alicelle olevansa Bob ja Bobille olevansa Alice. Esimerkiksi klassisen ilman autentikaatiota toimivan (anonyymin) Diffie-Hellman -protokollan tapauksessa Eve jakaisi molempien osapuolten välillä oman salaisen eksponenttinsa ja näin ollen pystyisi lukemaan salatut viestit molempiin suuntiin. Tätä tilannetta on havainnollistettu kuvassa 4.



Kuva 4. Man in the middle -hyökkäys anonyymiä Diffie-Hellman -protokollaa vastaan.

Tämän ongelman ratkaisemiseksi on kehitetty menetelmä nimeltään autentikaatiokoodit (message authentication codes), sekä julkisen avaimen kryptosysteemeiden kohdalla digitaaliset allekirjoitukset (digital signatures).

Autentikaatiokoodi muodostetaan yksityisen avaimen ja selkokielisen tekstin pohjalta $T_1 = \mathbf{MAC}(K, M)$, missä K on salainen avain ja M salattava viesti, ja lähetetään sitten viestin mukana. Vastaanottaja laskee uudestaan $T_2 = \mathbf{MAC}(K, M)$ ja vertailee koodeja keskenään. Jos $T_1 = T_2$, voi vastaanottaja olla varma sekä lähettäjästä, että datan

integriteetistä. Autentikaatiokoodeissa käytettävä salainen avain K on identtinen molemmilla osapuolilla, mikä tarkoittaa sitä, että avain on jouduttu jakamaan etukäteen. Kyseessä on siis symmetrinen systeemi. Autentikaatiokoodien muodostamiseen salauksen yhteydessä on olemassa kolme eri tapaa. Ensimmäisessä (Encrypt-and-MAC) selkokielinen teksti salataan ja selkokielisen tekstin pohjalta generoidaan koodi. Nämä molemmat lähetetään vastaanottajalle. Tämän menetelmän heikkous on siinä, että **MAC**-funktio voi teoriassa paljastaa tietoja selkokielisestä tekstistä, sillä **MAC**-funktion ei yleisesti ottaen tarvitse näyttää satunnaiselta. Toinen tapa (MAC-then-encrypt) on muodostaa ensin koodi T, sitten salata sekä selkokielinen teksti M, että koodi T yhdessä muodostaen salatun tekstin $C = \mathbf{E}(K, M \,||\, T)$, missä C on salattu teksti, $\mathbf{E}$ valittu salausalgoritmi ja $M \,||\, T$ selkokielinen teksti, johon on lisätty loppuun koodi T. Nyt vastaanottajalle lähetetään pelkästään C, jonka purettuaan hän voi verifioida viestin tagin avulla. Tämä tapa on turvallisempi kuin ensimmäinen tapa, sillä nyt selkokielisen tekstin pohjalta luotu koodi on salattu, jolloin se ei voi vuotaa tietoja selkokielisestä tekstistä. Kolmas tapa (Encrypt-then-MAC) on salata ensin selkokielinen teksti ja sitten luoda koodi salatusta tekstistä. Tämä on paras tapa, sillä nyt vastaanottajan tarvitsee vain laskea $T = \mathbf{MAC}(C)$ ja verrata sitä mukana tulleeseen tagiin. Jos tagit eroavat toisistaan, ei vastaanottajan tarvitse purkaa salattua tekstiä, vaan se voidaan saman tien jättää käsittelemättä. Tässä tavassa on myös se etu, että hyökkääjän pitää murtaa **MAC**-funktio voidakseen lähettää vastaanottajalle ”vihamielistä” dataa. [Aumasson, 2018]

Digitaaliset allekirjoitukset ratkaisevat saman ongelman kuin edellä mainitut autentikaatiokoodit, mutta näiden lisäksi ne tarjoavat ominaisuuden, missä allekirjoituksen alkuperäinen tekijä on varmasti kyseisen allekirjoituksen yksityisen avaimen hallussapitäjä. Avain, millä autentikaatiokoodit generoidaan, on symmetrinen lähettäjän ja vastaanottajan välillä, jolloin myös vastaanottaja pystyy generoimaan tismalleen saman autentikaatiokoodin. Digitaalisten allekirjoitusten osalta allekirjoitus generoidaan yksityisen avaimen pohjalta, mutta varmennetaan julkisella avaimella. Näin ollen voidaan olla lähes varmoja, että allekirjoituksen luoja on varmasti yksityisen avaimen hallussapitäjä (ellei hän ole jakanut avainta itse eteenpäin). [Aumasson, 2018]

Eräs menetelmä allekirjoittaa viestejä RSA:n avulla on RSA Probabilistic Signature Scheme (PSS), missä allekirjoitettavasta viestistä M luodaan tarkiste **Hash**(M) ja siihen lisätään satunnaista täytettä (salt) jollain algoritmilla. Tämän jälkeen täytettä sisältävä tarkiste P salataan RSA:n avulla siten, että P muutetaan kokonaisluvuksi $x < n$ ja tästä lasketaan $y = x^d \bmod n$, jolloin y on viestin M allekirjoitus. [Aumasson, 2018]

Autentikoitu Diffie-Hellman -protokolla korjaa edellä mainitun anonyymin Diffie-Hellmanin man in the middle -hyökkäyksen ongelman käyttämällä digitaalisia

allekirjoituksia. Näitä ei kuitenkaan generoida Diffie-Hellman -funktiolla $g^a \bmod p$, vaan käytetään esimerkiksi edellä mainittua PSS-menetelmää. [Aumasson, 2018]

Elliptisillä käyrillä toteutettu digitaalinen allekirjoitus (elliptic curve digital signature algorithm, ECDSA) taas toimii siten, että viestistä *M* luodaan jälleen tarkiste *h* = **Hash**(*M*) joka tulkitaan kokonaisluvuksi väliltä [0, *n*-1]. Tämän jälkeen valitaan satunnainen luku *k* väliltä [1, *n*-1] ja lasketaan elliptisen käyrän piste *k*G = (*x*, *y*). Sitten lasketaan

$$r = x \bmod n$$

ja

$$s = \frac{h + rd}{k} \bmod n$$

jolloin allekirjoitus muodostuu arvoista (*r*, *s*). Elliptisten käyrien avulla toteutettujen digitaalisten allekirjoitusten varmentaminen on hieman hitaampaa kuin RSA:lla, mutta allekirjoitusten pituus on moninkertaisesti pienempi ja allekirjoitusten luonti on moninkertaisesti nopeampi. Tästä syystä ECDSA on korvannut RSA:n digitaalisissa allekirjoituksissa monissa paikoissa ja se on esimerkiksi ainoa allekirjoitusalgoritmi Bitcoin-kryptovaluutassa. [Aumasson, 2018]

3. Kvanttitietokoneet

Kvanttitietokoneet perustuvat kvanttimekaniikan ilmiöihin, kuten superpositioon. Tässä luvussa avataan kvanttitietokoneiden eroavaisuuksia klassisiin tietokoneisiin, kvanttialgoritmien eroavaisuuksia klassisiin algoritmeihin sekä selvitetään kvanttitietokoneiden nykytilaa eli kuinka pitkällä ollaan kvanttitietokoneiden kehityksessä.

3.1. Ero klassiseen tietokoneeseen

Klassisessa tietokoneessa informaation perusyksikkö on bitti, joka voi olla joko tilassa 0 tai 1. Kvanttitietokoneessa perusyksikön nimi on kubitti (kvanttibitti, quantum bit, qubit), jolla on erikoinen ominaisuus: kubitti voi olla missä tahansa tilassa 0 ja 1 välillä, kunnes kubitin tila mitataan. Tällöin kubitin tila *luhistuu* klassiseksi tilaksi, joko 0 tai 1. Kubitin tilaa ennen mittausta kutsutaan *superpositioksi* ja sitä merkitään

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

missä $|\psi\rangle$ on *Dirac-notaatiolla* kirjoitettu kubitin symboli ja α ja β ovat todennäköisyysamplitudeja, jotka ovat kompleksilukuja ja kertovat todennäköisyydestä mitata 0 tai 1. Kubitin *magnitudi* eli todennäköisyys mitata joko 0 tai 1 on $|\alpha|^2$ tai $|\beta|^2$, missä

$$|\alpha| = |a + b\mathrm{i}| = \sqrt{a^2 + b^2}$$

ja luonnollisesti

$$|\alpha|^2 + |\beta|^2 = 1,$$

koska todennäköisyyksien summan pitää aina olla 1. Kubitti voi esimerkiksi olla superpositiossa

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

jolloin todennäköisyys mitata 0 tai 1 on yhtä suuri:

$$\left|\frac{1}{\sqrt{2}}\right|^2 = \sqrt{\frac{1}{\sqrt{2}}^2 + 0^2}^2 = \sqrt{\frac{1}{2}}^2 = \frac{1}{2}.$$

Tätä tilaa merkitään myös symbolilla $|+\rangle$. [Nielsen & Chuang, 2010]

Klassisessa tietokoneessa bittien tilaa manipuloidaan *loogisilla porteilla* (logic gates), joita ovat esimerkiksi AND, OR ja NOT -portit. AND-portti ottaa sisään kaksi bittiä ja toimii siten, että jos molempien bittien arvo on 1 on myös ulos tuleva bitti 1, muissa tapauksissa tulos on 0. OR-portti ottaa myös sisään kaksi bittiä ja ulos tulee 1, jos vähintään toinen bitti on tilassa 1, muissa tapauksissa 0. NOT-portti on yhden bitin portti, joka kääntää sisään tulevan bitin arvon toiseksi eli $0 \rightarrow 1$ ja $1 \rightarrow 0$. Myös kvanttitietokoneissa on loogisia portteja ja niitä voidaan matemaattisesti esittää matriiseina, joille pätee ainoastaan yksi sääntö: matriisin on oltava *unitäärinen*, eli matriisin A ja sen *liittomatriisin* $\mathrm{A}^{\dagger}$ tulon on oltava *identiteettimatriisi* $\mathrm{A}^{\dagger}\mathrm{A} = \mathrm{I}$. Liittomatriisi tai adjungoitu matriisi on matriisin A kompleksikonjugoidun matriisin transpoosi $A^{\dagger} = \bar{A}^{T}$. Tässä siis ensin korvataan jokainen matriisin alkio $z = a + bi$ sen kompleksikonjugaatilla $\overline{z} = a - bi$, jonka jälkeen transponoidaan kyseinen matriisi.

Eräs hyödyllisimmistä kvanttiporteista on yhden kubitin *Hadamardin portti*

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

joka vaihtaa kubitin tilan deterministisestä tilasta probabilistiseen tilaan

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

eli Hadamardin portti luo superposition. Tämä portti on validi kvanttiportti, sillä

$$H^{\dagger} = H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

jolloin

$$H^{\dagger}H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Hadamardin muunnos on *m*:n kubitin Hadamardin portti, joka on määritelty rekursiivisesti Hadamardin portin avulla seuraavasti

$$H_0 = 1$$

$$H_m = \frac{1}{\sqrt{2}}\begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$$

missä $m > 0$. Nyt esimerkiksi

$$H_1 = H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

ja

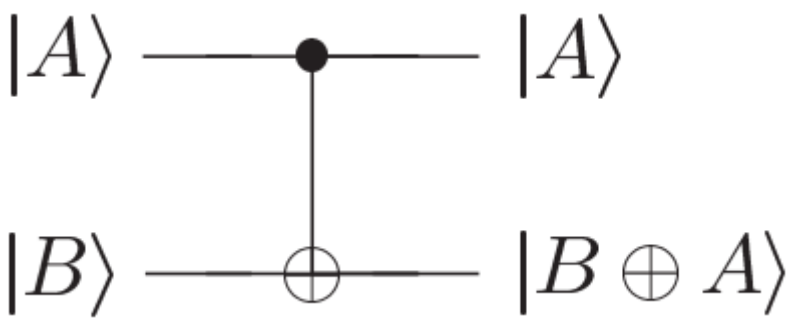
$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Klassisella tietokoneella on myös olemassa NAND-portti, jonka totuustaulu esitetään taulukossa 3.

p	**q**	**p NAND q**
0	0	1
0	1	1
1	0	1
1	1	0

Taulukko 3. NAND-portin totuustaulu.

Tähän porttiin liittyy todistus, jossa osoitetaan, että jokainen funktio biteille on mahdollista implementoida käyttämällä pelkästään NAND-portteja. NAND-portti on täten *universaali* portti. Vastaava universaali portti kvanttitietokoneilla saadaan CNOT-portilla sekä yhden kubitin porteilla. CNOT-portin piiriesitys nähdään kuvassa 5.



Kuva 5. CNOT-portin piirikuvaus. $|A\rangle$ on kontrollikubitti ja $|B\rangle$ kohdekubitti. Vaakaviivat esittävät ajan kulkua eteenpäin. Pystyviiva kuvaa ehdollista NOT-

operaatiota, jossa suoritetaan XOR-operaatio kontrolli- ja kohdekubitin välillä ja tallennetaan tulos kohdekubittiin. [Nielsen & Chuang, 2010]

Kuvan 5 XOR-operaation voi myös tulkita yhteenlaskuna modulo 2. Tilan muutos voidaan kirjoittaa eksplisiittisesti

$$|00\rangle \xrightarrow{U_{CN}} |00\rangle$$
$$|01\rangle \xrightarrow{U_{CN}} |01\rangle$$
$$|10\rangle \xrightarrow{U_{CN}} |11\rangle$$
$$|11\rangle \xrightarrow{U_{CN}} |10\rangle.$$

CNOT-portin matriisiesitys on

$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Mainittujen kvanttiporttien lisäksi on monia muitakin kvanttiportteja, sillä määritelmän mukaan niitä voi olla ääretön määrä. Näiden porttien avulla rakennetaan kaikki kvanttialgoritmit.

3.2. Kvanttialgoritmit

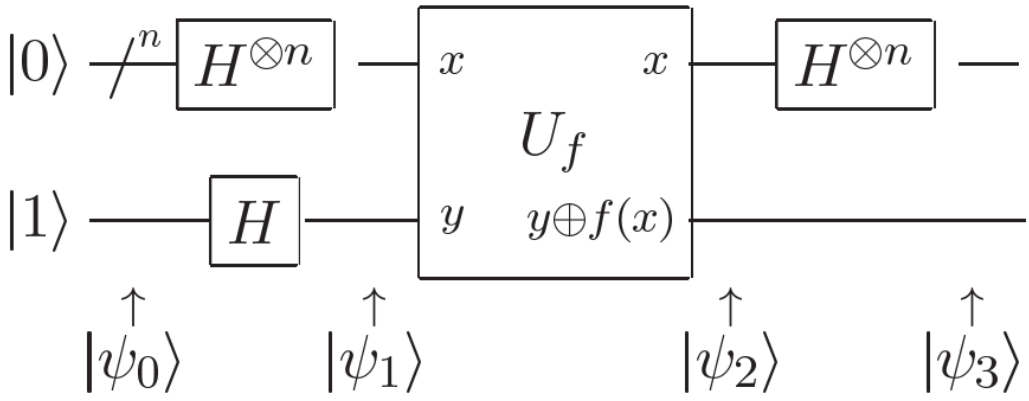
Kvanttialgoritmit eroavat klassisilla tietokoneilla suoritettavista algoritmeista siten, että emme voi saada superpositiossa olevasta kubitista mitään tietoa, sillä sen tilan mittaaminen tuhoaa superposition. Kvanttialgoritmit perustuvat superpositiossa olevien kubittien tilan transformoimiseen kvanttiporteilla tarkoituksena muuttaa mittaustulosten todennäköisyyksiä ja näin saada mittaustodennäköisyydet sellaiseen tilaan, että mittaamisen jälkeen saamme varmuudella ratkaisun ongelmaan.

Tässä luvussa tarkastellaan lähemmin kolmea tunnettua kvanttialgoritmia: Deutsch-Jozsa -algoritmia, Groverin hakualgoritmia ja Shorin algoritmia. Deutsch-Jozsa -algoritmi on yksinkertaisin kvanttialgoritmi, jolla pyritään demonstroimaan, miten kvanttitietokoneella voidaan ratkaista tietyn tyyppisiä ongelmia huomattavasti vähäisemmällä määrällä operaatioita verrattuna klassiseen tietokoneeseen. Varsinaisia käyttötapauksia kyseisellä algoritmilla ei ole. Groverin hakualgoritmilla pystytään löytämään n:n alkion järjestämättömästä hakuavaruudesta haluttu alkio aikakompleksisuudella $O(\sqrt{n})$, missä klassisella tietokoneella saman haun tekeminen vaatii aikakompleksisuuden $O(n)$. Täten Groverin algoritmi mahdollistaisi esimerkiksi 128-bittisen salausavaimen brute-force -haun 2^{64} operaatiolla 2^{128}:n sijasta. Shorin

algoritmilla taas etsitään funktion jakso eksponentiaalisesti nopeammin kuin klassisella tietokoneella on mahdollista. Shorin algoritmin aikakompleksisuus on luokkaa $O((log\ N)^2(log\ log\ N)(log\ log\ log\ N))$, missä N on tekijöihinsä jaettava kokonaisluku, kun vastaavasti nopeimman klassisella tietokoneella saman ongelman ratkaisevan algoritmin *yleisen lukukuntaseulan* (general number field sieve) aikakompleksisuus on $O(e^{1.9(log\ N)^{\frac{1}{3}}(log\ log\ N)^{\frac{2}{3}}})$. Funktion jakson avulla ratkaistaan RSA:n taustalla oleva kokonaislukujen tekijöihinjaon ongelma sekä Diffie-Hellmanin ja elliptisten käyrien Diffie-Hellmanin taustalla oleva diskreetin logaritmin ongelma. Toisin sanottuna Shorin algoritmi mahdollistaa näiden julkisen avaimen kryptosysteemien osalta yksityisen avaimen selvittämistä julkisesta avaimesta.

3.2.1. Deutsch-Jozsa -algoritmi

Deutsch-Jozsa -algoritmi on David Deutschin ja Richard Jozsan vuonna 1992 kehittämä kvanttialgoritmi, jolla demonstroidaan yksinkertaista ongelmaa, mihin kvanttialgoritmi tarjoaa eksponentiaalisesti nopeamman ratkaisun pahimmassa tapauksessa, kuin klassinen tietokone [Deutsch & Jozsa, 1992]. Ongelma on nimeltään Deutschin ongelma. Ongelman kuvaus on seuraava: on olemassa funktio ***f***, joka on joko vakiofunktio tai tasainen funktio. Vakiofunktiossa ***f***(x) = 0 tai ***f***(x) = 1 jokaiselle arvolle x, joka on kokonaisluku väliltä [1, 2^n]. Tasaisessa funktiossa ***f***(x) = 0 puolelle arvoista x ja ***f***(x) = 1 toiselle puolelle. Kuinka monta kertaa funktiota ***f*** tulee kutsua, jotta tiedämme varmuudella, onko kyseessä vakiofunktio vai tasainen funktio? Klassisella tietokoneella tarvitsemme pahimmassa tapauksessa vähintään $2^{n-1}+1$ funktiokutsua, jos ensimmäiset 2^{n-1} kutsua palauttavat kaikki 0 ja vasta viimeisellä kutsulla saamme tuloksen 1. Kvanttitietokoneella tarvitsemme vain yhden kutsun, jonka jälkeen tiedämme varmuudella, onko kyseessä vakio- vai tasafunktio. Kuvassa 6 esitetään Deutsch-Jozsa -algoritmin toteutus piiriesityksenä.



Kuva 6. Deutsch-Jozsa -algoritmin piirikuvaus, missä merkintä $/^n$ tarkoittaa *n*-määrää kubitteja, $H^{\otimes n}$ Hadamardin muunnosta *n*:lle kubitille ja U_f ”oraakkelia”, joka evaluoi ensimmäisen rekisterin arvolla annetun funktion ***f*** toiseen rekisteriin eli hoitaa muunnoksen $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$ [Nielsen & Chuang, 2010].

Algoritmi alkaa siten, että *n* kubittia on tilassa $|0\rangle$ ja *n* + 1:s kubitti tilassa $|1\rangle$:

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$$

Tämän jälkeen jokainen kubitti kulkee Hadamardin portin läpi, jolloin kubitit ovat superpositiossa:

$$|\psi_1\rangle = \sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

eli ensimmäiset *n* kubittia ovat tasaisessa jokaisen *x*:n arvon superpositiossa ja *n* + 1:s kubitti on tasaisessa $|0\rangle$ ja $|1\rangle$ superpositiossa. Nyt suoritetaan funktiokutsu ***f***, eli kvanttiportin mukainen transformaatio $U_f : |x, y\rangle \rightarrow |x, y \oplus f(\mathrm{x})\rangle$, jonka jälkeen tila on

$$|\psi_2\rangle = \sum_{x \in \{0,1\}^n} \frac{|x\rangle|f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

Nyt jos tarkastellaan mahdollisia ***f***:n arvoja 0 tai 1, saadaan seuraavat vaihtoehdot tilalle:

$$\boldsymbol{f}(x) = 0 \rightarrow \frac{|x\rangle|0\rangle - |x\rangle|1\rangle}{\sqrt{2^{n+1}}} = \frac{|x\rangle}{\sqrt{2^n}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$\boldsymbol{f}(x) = 1 \rightarrow \frac{|x\rangle|1\rangle - |x\rangle|0\rangle}{\sqrt{2^{n+1}}} = \frac{|x\rangle}{\sqrt{2^n}} \frac{|1\rangle - |0\rangle}{\sqrt{2}}$$

Tästä huomataan, että jälkimmäisen tilan voi kirjoittaa myös

$$\frac{|x\rangle}{\sqrt{2^n}}\frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\frac{|x\rangle}{\sqrt{2^n}}\frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

jolloin saadaan molemmille vaihtoehdoille tiivistetympi muoto

$$|\psi_2\rangle = \sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Tästä eteenpäin viimeistä kubittia ei enää tarvita. Lopulta ensimmäiset *n* kubittia lähetetään jälleen Hadamardin porttien läpi:

$$H^{\otimes n}\left(\sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}}\right) = \sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}}{\sqrt{2^n}} H^{\otimes n}|x\rangle$$

Hadamardin muunnos $H^{\otimes n}|x\rangle$ voidaan myös kirjoittaa muodossa

$$H^{\otimes n}|x\rangle = \sum_{z \in \{0,1\}^n} \frac{(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}}{\sqrt{2^n}}|z\rangle$$

missä $\boldsymbol{x} \cdot \boldsymbol{z}$ kuvaa $\boldsymbol{x}$:n ja $\boldsymbol{z}$:n bittiesitysten pistetuloa modulo 2. Jos esimerkiksi

$$\boldsymbol{x} = |3\rangle = |011\rangle$$
$$\boldsymbol{z} = |5\rangle = |101\rangle$$

niin

$$\boldsymbol{x} \cdot \boldsymbol{z} = 0 + 0 + 1 \bmod 2 = 1.$$

Nyt tila $|\psi_3\rangle$ voidaan kirjoittaa muodossa

$$|\psi_3\rangle = \sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} \frac{(-1)^{\boldsymbol{x}\cdot\boldsymbol{z}}}{\sqrt{2^n}}|z\rangle$$

Tästä saadaan tiiviimpi muoto

$$|\psi_3\rangle = \sum_{z \in \{0,1\}^n} \sum_{x \in \{0,1\}^n} \frac{(-1)^{\boldsymbol{x \cdot z + f(x)}}}{2^n} |z\rangle$$

Tästä yhtälöstä huomataan, että tilan $|0\rangle^{\otimes n}$, missä kaikki kubitit mitattaisiin nollaksi, amplitudi olisi

$$\sum_{x \in \{0,1\}^n} \frac{(-1)^{\boldsymbol{f}(x)}}{2^n}$$

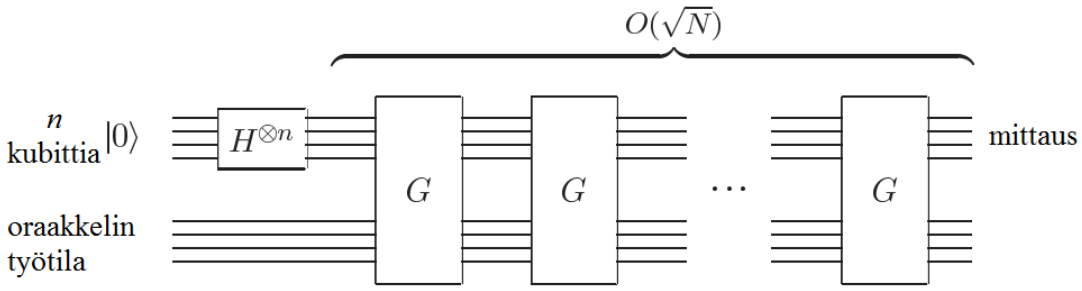
ja näin ollen ***f***:n ollessa vakiofunktio mitataan tila $|0\rangle^{\otimes n}$ huolimatta ***f***:n arvosta (0 tai 1). Jos taas ***f*** olisi tasainen, mittauksen jälkeen saamme tulokseksi jotain muuta kuin pelkkiä nollia.

Kuten edellä todettiin, tämä ongelma ei ole mitenkään erityisen tärkeä, mutta sen ratkaisu demonstroi kuinka kvanttitietokone pystyy ratkaisemaan tietyn tyyppisiä ongelmia nopeammin kuin klassinen tietokone. Algoritmin ytimessä on funktion ***f*** evaluointi superpositiossa olevilla kubiteilla, mikä tarkoittaa käytännössä yhtä ***f***(*x*) kutsua kaikilla mahdollisilla *x*:n arvoilla. Tätä kutsutaan *kvanttirinnakkaisuudeksi* (quantum parallelism) ja se on monen kvanttialgoritmin ytimessä [Nielsen & Chuang, 2010].

3.2.2. Groverin hakualgoritmi

Groverin hakualgoritmi on Lov Groverin [1996] kehittämä kvanttialgoritmi, joka hakee *N*:n alkion joukosta alkion *x*, jolle ***f***(*x*) = 1 ja kaikille muille alkioille ***f***(*x*) = 0 ajassa $O(\sqrt{N})$. Klassisella tietokoneella saman ongelman ratkaiseminen vie $O(N)$ operaatiota, sillä vasta viimeinen alkio joukosta saattaa olla haluttu alkio. Grover kuvasi alun perin algoritminsa järjestämättömän tietokannan hakualgoritmina, mutta kryptografian osalta vaikutus tulee siitä, että funktio ***f*** voi olla esimerkiksi salausfunktio, jolla luodaan yksityinen avain symmetrisessä kryptosysteemissä. Täten siis Groverin algoritmilla voi löytää 128-bittisen salausavaimen $\sqrt{2^{128}} = 2^{64}$ operaatiolla, mikä ei olekaan enää mahdotonta järkevässä ajassa. [Aumasson, 2018]

Groverin algoritmin piirikuva nähdään kuvassa 7.

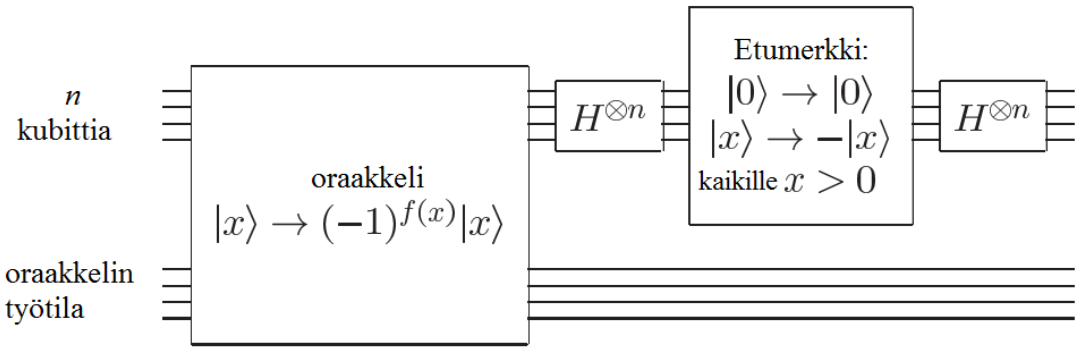


Kuva 7. Groverin algoritmin piirikuvaus, missä G kuvaa yhtä Groverin iteraatiota. [Nielsen & Chuang, 2010]

Algoritmi toimii siten, että $n = \log_2 N$ kubittia alustetaan tasaisesti superpositioon, eli tilaan

$$\sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}}$$

missä $|x\rangle$ esittää kaikkia mahdollisia arvoja, joita n kubitilla voi esittää. Seuraavaksi suoritetaan ns. *Groverin iteraatio* G peräkkäin $O(\sqrt{N})$ kertaa, jolloin halutun hakutuloksen amplitudi kasvaa aina lähemmäs lukua 1, kun taas muiden arvojen amplitudi pienenee. Groverin iteraation piirikuva nähdään kuvassa 8.



Kuva 8. Groverin iteraatio, missä oraakkeliportti vaihtaa halutun hakutuloksen amplitudin etumerkin ja etumerkkiportti vaihtaa kaikkien nollasta poikkeavien arvojen etumerkin. Oraakkeli käyttää implementaatiossaan myös lisäkubitteja omassa työtilassaan, mutta ne eivät ole tärkeitä algoritmin ymmärryksen kannalta. [Nielsen & Chuang, 2010]

Esimerkkinä tarkastellaan kahden kubitin tilannetta, jolloin hakualue $N = 4$ ja mahdolliset luvut ovat $|00\rangle = 0$, $|01\rangle = 1$, $|10\rangle = 2$ ja $|11\rangle = 3$. Olkoon haluttu hakutulos arvo $|01\rangle = 1$:

1. Luodaan superpositio: $|00\rangle \xrightarrow{H^{\otimes 2}} \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$
2. Suoritetaan oraakkelikutsu: $\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \xrightarrow{O} \frac{|00\rangle - |01\rangle + |10\rangle + |11\rangle}{2}$
3. Hadamardin muunnos: $\frac{|00\rangle - |01\rangle + |10\rangle + |11\rangle}{2} \xrightarrow{H^{\otimes 2}} \frac{|00\rangle + |01\rangle - |10\rangle + |11\rangle}{2}$
4. Etumerkin muunnos: $\frac{|00\rangle + |01\rangle - |10\rangle + |11\rangle}{2} \xrightarrow{P} \frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2}$
5. Hadamardin muunnos: $\frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2} \xrightarrow{H^{\otimes 2}} |01\rangle$

Viimeisen askeleen jälkeen kaikkien muiden arvojen amplitudi on 0 paitsi halutun hakutuloksen $|01\rangle = 1$, jonka amplitudi on 1. Askeleet 2 – 5 muodostavat yhden Groverin iteraation, jota piti kahden kubitin tilanteessa toistaa vain yhden kerran.

Visuaalisesti Groverin algoritmia voi ajatella niin, että oraakkelikutsu kääntää halutun hakutuloksen amplitudin etumerkin, mutta sen magnitudi eli mittauksen todennäköisyys pysyy samana. Kaikkien muiden ei-haluttujen hakutulosten amplitudi pysyy samana. Tämän jälkeen Hadamardin muunnosten ja etumerkkiportin yhteisvaikutus kääntää kaikkien mahdollisten hakutulosten amplitudit systeemin keskiarvoamplitudin ympäri. Kahden kubitin tilanteessa aluksi kaikkien mahdollisten tulosten amplitudi on $\frac{1}{2}$ ja magnitudi $(\frac{1}{2})^2 = \frac{1}{4}$. Oraakkelikutsun jälkeen tilan $|01\rangle = 1$ amplitudi on $-\frac{1}{2}$, mutta magnitudi edelleen $(-\frac{1}{2})^2 = \frac{1}{4}$. Hadamardin muunnosten ja etumerkin muunnoksen jälkeen kaikkien mahdollisten hakutulosten amplitudit käännetään systeemin keskiarvoamplitudin ympäri. Keskiarvo on tässä tapauksessa $\frac{1}{4}$, jolloin käännettäessä amplitudit $\frac{1}{2}$ tämän keskiarvon ympäri, päätyvät ne arvoon 0. Käännettäessä amplitudi $-\frac{1}{2}$ keskiarvon $\frac{1}{4}$ ympäri, päädytään arvoon 1. Tästä tilan muutoksesta on muun muassa Sam Wehnerin toimesta piirretty erinomainen visualisaatio [Wehner, 2017].

3.2.3. Shorin algoritmi

Shorin algoritmi on kvanttitietokoneella kehitetyistä algoritmeista kaikista merkityksellisin kryptografian kannalta. Tämän algoritmin avulla pystytään murtamaan kaikki julkisen avaimen kryptosysteemit, jotka perustuvat kokonaislukujen tekijöihinjaon ongelmaan (RSA), tai diskreetin logaritmin ongelmaan (DH, ECDH) selvittämällä yksityinen avain julkisesta avaimesta. Algoritmi lähtee siitä oletuksesta, että edellä mainitut ongelmat pystytään redusoimaan funktion jakson löytämisen ongelmaksi (period finding problem).

Tarkastellaan ensin RSA:ta ja kokonaislukujen tekijöihinjakoa. Kuten luvussa 2.3 todettiin, RSA:n julkisen avaimen mukana kulkee kokonaisluku $N = pq$, missä p ja q ovat suuria alkulukuja. Salattu teksti on muotoa $x^e \bmod N$. Koska

$$\boldsymbol{f}(x) = x^e \bmod N$$

on olemassa kokonaisluku *r*, jolle pätee

$$\boldsymbol{f}(x) = \boldsymbol{f}(x + r)$$

Tällaista funktiota kutsutaan *jaksolliseksi funktioksi*, jonka jakso on *r*. Esimerkiksi funktion $\boldsymbol{f}(\mathrm{x}) = x^3 \bmod 5$ jakso voidaan laskea käsin:

$$\boldsymbol{f}(1+0) = 1^3 \bmod 5 = 1$$
$$\boldsymbol{f}(1+1) = 2^3 \bmod 5 = 3$$
$$\boldsymbol{f}(1+2) = 3^3 \bmod 5 = 2$$
$$\boldsymbol{f}(1+3) = 4^3 \bmod 5 = 4$$
$$\boldsymbol{f}(1+4) = 5^3 \bmod 5 = 0$$
$$\boldsymbol{f}(1+5) = 6^3 \bmod 5 = 1$$

Näin ollen funktion $\boldsymbol{f}$ jakso on 5, sillä $\boldsymbol{f}(\mathrm{x}) = \boldsymbol{f}(\mathrm{x}+5)$.

Shorin algoritmissa on sekä klassisella, että kvanttitietokoneella suoritettavia vaiheita. Koska ei ole syytä suorittaa kvanttitietokoneella vaiheita, jotka voi suorittaa klassisella, suoritetaan vain funktion jakson haun tekevä askel kvanttitietokoneella [Shor 1996]. Algoritmissa käytetään sisäisesti myös *Eukleideen algoritmia* suurimman yhteisen tekijän löytämiseksi. Eukleideen algoritmi löytää kokonaislukujen *n* ja *m*, missä $n > m$ suurimman yhteisen tekijän $\boldsymbol{syt}(n, m)$ seuraavaa kaavaa toistamalla, kunnes jakojäännös $r_n = 0$:

$$n = q_0 m + r_0$$
$$m = q_1 r_0 + r_1$$
$$r_0 = q_2 r_1 + r_2$$
$$r_1 = q_3 r_2 + r_3$$
$$\vdots$$

Kun jakojäännös $r_n = 0$, on viimeistä edellinen jakojäännös r_{n-1} lukujen *n* ja *m* suurin yhteinen tekijä. Esimerkiksi lukujen $n = 38$ ja $m = 16$ suurin yhteinen tekijä saadaan seuraavasti:

$$38 = 2 \cdot 16 + 6$$
$$16 = 2 \cdot 6 + 4$$
$$6 = 1 \cdot 4 + 2$$
$$4 = 2 \cdot 2 + 0$$

Nyt $r_3 = 0$, jolloin lukujen 38 ja 16 suurin yhteinen tekijä on $r_2 = 2$.

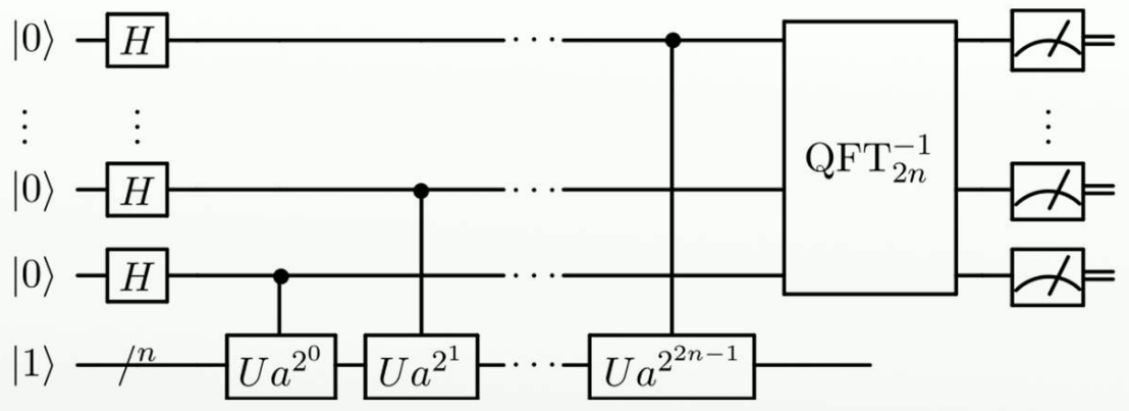
Shorin algoritmin vaiheet ovat seuraavat, kun halutaan etsiä luvun *N* kokonaislukutekijät:

1. Valitaan satunnainen luku $a < N$
2. Testataan, onko *a* *N*:n ei-triviaali kokonaislukutekijä laskemalla suurin yhteinen tekijä $\boldsymbol{syt}(a, N)$ Eukleideen algoritmilla
    a. Jos $\boldsymbol{syt}(a, N) \neq 1$, *a* on ei-triviaali *N*:n tekijä, jolloin olemme valmiit.
    b. Muussa tapauksessa etsitään funktion $\boldsymbol{f}(x) = a^x \bmod N$ jakso *r*.
3. Jos jakso *r* on pariton luku tai jos $a^{r/2} \equiv -1 \pmod N$, palataan askeleeseen 1.
4. Muussa tapauksessa $\boldsymbol{syt}(a^{r/2} + 1, N)$ ja $\boldsymbol{syt}(a^{r/2} - 1, N)$ ovat molemmat *N*:n ei-triviaaleja tekijöitä, joten algoritmin suoritus päättyy.

Askeleen 2 kohta b suoritetaan kvanttitietokoneella tai kvanttiprosessorilla. Tarvittava kubittien määrä riippuu luvusta *N*, jonka tekijät koitetaan löytää. Kubittien määrän tulisi olla *q*, missä

$$N^2 \leq 2^q < 2N^2 \text{ [Shor, 1996]}$$

Tämän vaiheen kuvaava kvanttipiiri nähdään kuvassa 9.



Kuva 9. Shorin algoritmin jaksonhakualirutiinin piirikuvaus. Portit $U_{a^{2^0}} - U_{a^{2^{2n-1}}}$ kuvaavat oraakkelin U_f, missä $\boldsymbol{f}(x) = x^e \bmod N$. Portti QFT_{2n}^{-1} kuvaa käänteistä

kvantti-Fourier-muunnosta ja mittausportit lopussa kuvaavat kubittien mittausta. [Nielsen & Chuang, 2010]

Algoritmissa käytettävä kvantti-Fourier-muunnos on käytännössä diskreetti Fourier-muunnos, joka operoi kubittien amplitudivektoreilla. $N = 2^n$ kubitin kvantti-Fourier-muunnoksen matriisiesitys on

$$QFT_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w_n & w_n^2 & \cdots & w_n^{N-1} \\ 1 & w_n^2 & w_n^4 & \cdots & w_n^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w_n^{N-1} & w_n^{2(N-1)} & \cdots & w_n^{(N-1)(N-1)} \end{bmatrix},$$

missä $w_n = e^{\frac{2\pi i}{2^n}}$ on n:nes yksikköjuuri. Esimerkiksi kahden kubitin tilan

$$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

amplitudivektori on

$$\frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

jolloin sen kvantti-Fourier-muunnos on

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Jaksonhakualirutiinin kvanttipiirissä käytetään kahta kvanttirekisteriä, joissa molemmissa on q (kuvassa n) kubittia aluksi tiloissa $|0\rangle^{\otimes q}|1\rangle^{\otimes q}$. Ensimmäinen rekisteri muunnetaan jälleen tasajakautuneeksi superpositioksi

$$\sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}}$$

Seuraavaksi suoritetaan oraakkelikutsu O_f, joka muuntaa tilan seuraavasti:

$$\sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} |1\rangle^{\otimes q} \overset{O_f}{\rightarrow} \sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} |\boldsymbol{f}(x)\rangle$$

Tämä vaihe evaluoi funktion $\boldsymbol{f}(x)$, joka tässä tapauksessa on $x^e \bmod N$ ja tallentaa tuloksen toiseen rekisteriin. Nyt meillä on tilanne, missä mittaamalla toisen rekisterin $\boldsymbol{f}(x)$ saamme jonkun luvun $\boldsymbol{f}(x_0)$, jolloin ensimmäinen rekisteri luhistuu $\boldsymbol{f}$:n alkukuvaan. Koska $\boldsymbol{f}$ on jaksollinen funktio, $\boldsymbol{f}(x_0)$:n alkukuva on

$$x_0 \in \left\{x_0, x_0 + r, x_0 + 2r, \ldots, x_0 + (\frac{N}{r} - 1)r\right\}$$

missä *r* on funktion jakso. Viimeinen vaihe ennen mittausta on suorittaa käänteinen kvantti-Fourier-muunnos. Kvantti-Fourier-muunnoksella on seuraava ominaisuus: Jos $\boldsymbol{f}$ on jaksollinen ja sen jakso on *r*, niin Fourier-muunnos $\hat{\boldsymbol{f}}$ on jaksollinen jaksolla $\frac{N}{r}$, missä *N* on funktion modulus. Fourier-muunnoksen jälkeen saamme lopulta mittauksesta jonkun monikerran $k\frac{N}{r}$. Tästä ei voida vielä varmuudella päätellä jaksoa *r*, joten algoritmia voidaan joutua toistamaan muutaman kerran, kunnes meillä on muutama monikerta $k_1\frac{N}{r}$, $k_2\frac{N}{r}$, joista sitten Eukleideen algoritmilla voidaan laskea suurimmaksi yhteiseksi tekijäksi $\frac{N}{r}$. Kun $\frac{N}{r}$ on tiedossa, saamme selville *r*:n *ketjumurtolukuesityksestä*. Ketjumurtolukuesityksessä rationaaliluku $\frac{a}{b}$ esitetään muodossa

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ldots}}}$$

Tämän esityksen *n*:s *konvergentti* on reaaliluku $\frac{d_n}{r_n}$, jossa luvut d_i ja r_i on määritelty seuraavasti:

$$d_0 = a_0, d_1 = a_0 a_1 + 1, d_n = a_n d_{n-1} + d_{n-2}$$
$$r_0 = 1, r_1 = a_1, r_n = a_n r_{n-1} + r_{n-2}.$$

Kun konvergentit on saatu ketjumurtolukuesityksestä, on jokin luvuista r_i funktion $\boldsymbol{f}$ jakso.

Etsitään esimerkiksi luvun $N = 91$ kokonaislukutekijät. Ensin on löydettävä kvanttipiirissä tarvittavien kubittien määrä. Shorin [1996] mukaan tarvitaan *q* kubittia, missä

$$N^2 \leq 2^q < 2N^2$$

Tässä tapauksessa

$$91^2 = 8\,281 \leq 2^q < 16\,562 = 2 \cdot 91^2$$

jolloin $q = 14$, sillä $2^{14} = 16\,384$. Seuraavaksi valitaan satunnainen luku $a < N$, joka on keskenään jaoton luvun *N* kanssa eli $\boldsymbol{syt}(a, N) = 1$. Esimerkiksi 15 on tällainen luku. Seuraavaksi etsitään funktion

$$\boldsymbol{f}(x) = 15^x \bmod 91$$

jakso kvanttipiirin avulla. Tietokoneohjelmalla testaamalla tämän funktion jaksoksi saadaan 12, joten tämä luku on saatava myös jaksonhakualirutiinilla:

1. Luodaan superpositio ensimmäiseen rekisteriin: $\frac{1}{\sqrt{2^{14}}}\sum_{x=0}^{2^{14}-1}|x\rangle|1\rangle^{\otimes n}$
2. Lasketaan ***f***(x) toiseen rekisteriin: $\frac{1}{\sqrt{2^{14}}}\sum_{x=0}^{2^{14}-1}|x\rangle|15^x \bmod 91\rangle$
3. Suoritetaan transformaatio QFT_{14}^{-1} ensimmäiseen rekisteriin, jolloin on lähes yhtäläinen todennäköisyys mitata luvut 0, ~1365, ~2730, …, ~16 384 - ~1365. Nämä luvut ovat luvun $\frac{16384}{12}$ monikertoja.
4. Mitataan ensimmäisen kvanttirekisterin kubitit käänteisessä järjestyksessä, kuten Shor [1996] julkaisussaan esitti, jolloin saadaan joku monikerroista $k_1\frac{16384}{12}$.

Tämä askel voidaan joutua toistamaan muutaman kerran, kunnes saadaan joko mitattua arvo 1365, joka on $\frac{2^q}{r}$, tai muutaman monikerran avulla niiden suurin yhteinen tekijä 1365. Nyt algoritmia voidaan jatkaa klassisen tietokoneen puolella, jolloin seuraava askel on laskea luvusta 1365 funktion jakso *r* esittämällä luku $\frac{1365}{16384}$ ketjumurtolukuesityksen avulla:

$$0 + \cfrac{1}{12 + \cfrac{1}{341 + \cfrac{1}{4}}}$$

Tästä esityksestä saadaan konvergentit:

$$d_0 = 0, d_1 = 1, d_2 = 341, d_3 = 1364$$

$$r_0 = 1, r_1 = 12, r_2 = 4093, r_3 = 16384$$

Nyt nähdään, että ainoa mahdollinen vaihtoehto funktion jaksolle on $r_i = 12$. Koska 12 on parillinen luku ja

$$15^6 \neq -1 \bmod 91$$

ovat luvut

$$\boldsymbol{syt}(15^6 + 1, 91) = 13$$
$$\boldsymbol{syt}(15^6 - 1, 91) = 7$$

luvun 91 tekijöitä, joten $91 = 13 \cdot 7$.

Diskreetin logaritmin ongelman ratkaiseminen kvanttitietokoneen avulla noudattaa samaa periaatetta kuin yllä esitetty kokonaisluvun tekijöihin jako. Tässä tapauksessa käytetään kolmea kvanttirekisteriä, joista kaksi ensimmäistä asetetaan superpositioon ja kolmanteen lasketaan funktio $\boldsymbol{f}(x_1, x_2)$, missä

$$\boldsymbol{f}(x_1, x_2) = g^{x_1} y^{x_2}$$

Tässä yhteydessä *g* ja *y* ovat Diffie-Hellman -protokollan yhteydessä olevasta yhtälöstä

$$y = g^x \bmod p$$

tiedossa olevat luvut *g* ja *y*. Funktio $\boldsymbol{f}$ on jaksollinen, sillä on olemassa luvut ω ja ω′, joille pätee

$$\boldsymbol{f}(a, b) = \boldsymbol{f}(a + \omega, b + \omega')$$

jokaiselle kokonaisluvulle *a* ja *b*. Tästä voidaan päätellä, että

$$g^{\omega} y^{\omega'} \bmod p = 1$$

Nyt voidaan sijoittaa *y*:n tilalle g^x, jolloin

$$g^{\omega + x\omega'} \bmod p = 1$$

Tämä tarkoittaa samaa kuin

$$\omega + x\omega' \bmod q = 0,$$

missä q on äärellisen ryhmän Z_p^* alkioiden määrä (order). Kun p on alkuluku, on $q = (p-1)$. Tästä siis saadaan $x = \frac{-\omega}{\omega'}$, jolloin funktion jakson löytämällä löydämme salaisen eksponentin x. [Aumasson, 2018]

3.3. Kvanttitietokoneiden tila

John Preskill [2011] määritteli *kvanttiylemmyyden* (quantum supremacy) hetkenä, jolloin kvanttitietokoneet pystyvät ratkaisemaan ongelmia, joita klassiset tietokoneet eivät pysty. Kryptografian kannalta tämä hetki on silloin kun Shorin algoritmin avulla voidaan murtaa esimerkiksi RSA. Nykypäivän RSA-avainten koko on kuitenkin yleensä 2048 bittiä, mikä tarkoittaa sitä, että kvanttitietokone vaatisi vähintään 2048 *lomittunutta* (entangled) kubittia. Nykyhetken ennätys lomittuneissa kubiteissa on 18 [Wang, 2018], kun vuonna 2010 ennätys oli 14 [Monz, 2010]. Kehitys on siis ollut varsin hidasta, sillä lomittuneiden kubittien hallinta on hyvin vaikeaa. Suurin Shorin algoritmilla tekijöihinsä jaettu kokonaisluku on $21 = 3 \cdot 7$ [Martín-López et al. 2012]. Tässä algoritmissa vähennettiin tarvittavien kubittien määrää kierrättämällä yhtä kubittia. Myös muut ovat yrittäneet parantaa Shorin algoritmia vähentämällä nimenomaan tarvittavien kubittien määrää ajoajan kustannuksella [Bernstein et al. 2017]. Tämäkin kehitystie tulee ottaa huomioon, kun lasketaan kvanttiylemmyyden ajankohtaa.

Kvanttitietokoneiden rakentaminen on ennen kaikkea valtava insinöörityön haaste. Kvanttitietokoneen rakentaminen on erityisen hankalaa siksi, että kubitteina toimivat erittäin pienet, elektronien tai fotonien kokoiset partikkelit. Kubittien superposition voi pitää vakaana vain erittäin kylmissä, lähellä absoluuttista nollapistettä olevissa lämpötiloissa ja kubitteihin vaikuttavat ympäristön häiriötekijät, kuten lämpö ja magneettikentät. [Aumasson, 2018]

Haasteista huolimatta IT-jättiläiset, kuten Google, IBM ja Intel ovat täyttä vauhtia rakentamassa kvanttitietokoneita ja/tai kvanttiprosessoreita. IBM julkaisi tammikuussa 2019 ensimmäisen kaupalliseen käyttöön tulevan 20 kubitin kvanttitietokoneensa nimeltä IBM Q System One [IBM quantum computer, 2019]. Google julkaisi kehittävänsä 72 kubitin prosessoria nimeltään Bristlecone maaliskuussa 2018, joka on ennätysmäärä kubitteja universaalissa kvanttitietokoneessa [Google Bristlecone, 2018]. Intel on myös julkaissut vuonna 2018 kehittävänsä 49 kubitin prosessoria nimeltään Tangle Lake [Intel Tangle Lake, 2018].

Nämä edellä mainitut prosessorit perustuvat kvanttiporttiarkkitehtuuriin ja ovat siis ”universaaleja” kvanttitietokoneita. Yritys nimeltään D-Wave on kehittänyt toisenlaiseen arkkitehtuuriin, *kvanttihehkutukseen* (quantum annealing) perustuvan kvanttitietokoneen nimeltään D-Wave 2000Q, joka sisältää huikeat 2000 kubittia [D-Wave 2000Q, 2017]. Tähän arkkitehtuuriin pohjautuvat kvanttitietokoneet eivät kuitenkaan ole yleisesti ajateltuna universaaleja kvanttitietokoneita, joten niitä ei voida suoraan verrata kvanttiportteihin perustuviin kvanttitietokoneisiin [D-Wave Quantum Annealing, 2016].

Kvanttitietokoneiden tulemisen myötä on tutkijoiden mielenkiinto siirtynyt vahvasti myös vaihtoehtoisiin kryptosysteemeihin, jotka pohjautuvat sellaisiin matemaattisiin ongelmiin, joiden ratkaisemiseen ei ole (vielä) keksitty klassisia algoritmeja nopeampia kvanttialgoritmeja.

3.4. Kvanttikryptografia

Kvanttikryptografialla (quantum cryptography, quantum key distribution) tarkoitetaan sellaisia kryptosysteemeitä, jotka hyödyntävät kvanttimekaniikan lainalaisuuksia. Kvanttikryptografia on eri asia kuin kvanttiresistantti kryptografia, jolla taas tarkoitetaan klassisella tietokoneella toteutettavia kryptosysteemeitä, jotka ovat nykytiedon valossa ”turvassa” kvanttitietokoneilta eli käytännössä Shorin algoritmilta. Kvanttikryptografia lähtee siitä ajatuksesta, että Alice ja Bob vaihtavat keskenään salaisen avaimen muodostamiseen tarvittavia superpositiossa olevia kubitteja julkisen, turvattoman kanavan välityksellä. Kubitin mittaaminen aiheuttaa superposition luhistumisen, jolloin Even mitatessa näitä matkalla olevia kubitteja aiheuttaisi hän häiriöitä järjestelmässä. Tämä häiriö on jälkeenpäin havaittavissa Alicen tai Bobin mitatessa kubitit, jolloin he voivat hylätä kyseiset kubitit ja käyttää loppuja salausavaimen muodostamiseen. Charles Bennett ja Gilles Brassard kehittivät ensimmäisen kvanttiavaimenvaihtoprotokollan nimeltään BB84 vuonna 1984. Tämän jälkeen on kehitetty monia muitakin protokollia, mutta perusidea on sama. Tarkempaa tietoa kvanttikryptografiasta ja muun muassa BB84-protokollasta voi lukea Gilles Van Asschen kirjasta ”Quantum Cryptography and Secret-Key Distillation”. [Van Assche, 2006]

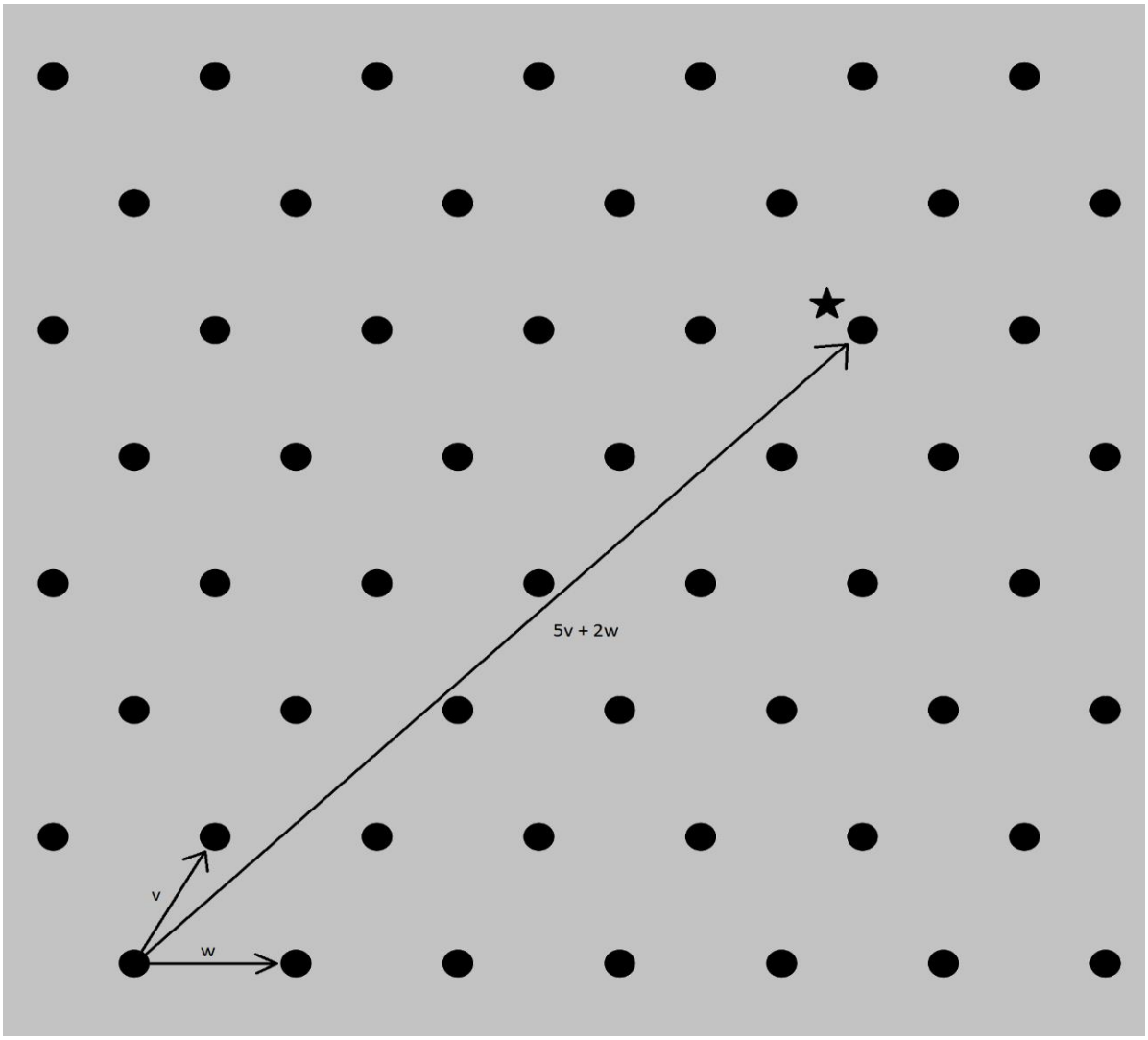
Kvanttikryptografia on mielenkiintoinen ajatus, sillä se tarjoaa todistettavasti varman turvallisuustason, koska se perustuu kvanttimekaniikan sääntöihin. Käytännön toteutus on kuitenkin vaikeaa, sillä informaation kantajina toimivia yksittäisiä fotoneja on vaikea luoda [Van Assche, 2006]. Kvanttikryptografia saattaa muutenkin osoittautua tarpeettomaksi, jos seuraavassa luvussa mainituista kvanttiresistanteista kryptosysteemeistä löytyy tarpeeksi hyvä korvaaja nykyisille kryptosysteemeille.

4. Kvanttiresistantti kryptografia

NIST aloitti vuonna 2017 prosessin, jonka tarkoituksena on standardisoida kvanttiresistantteja kryptosysteemeitä [Aumasson, 2018]. Puhutaan siis kryptosysteemeistä, jotka perustuvat sellaisiin matemaattisiin ongelmiin mihin ei ole vielä keksitty nopeampia ratkaisuja kvanttitietokoneilla suhteessa klassisiin tietokoneisiin. Tässä luvussa käydään lyhyesti läpi neljäntyyppisiä kvanttiresistantteja kryptosysteemeitä: hilaongelmiin pohjautuvia, koodipohjaisia, monimuuttujapohjaisia sekä hajautuspohjaisia kryptosysteemeitä.

4.1. Hilapohjainen kryptografia

Hila on käytännössä matemaattinen struktuuri, joka koostuu jaksollisen rakenteen omaavista pisteistä *n*-ulotteisessa avaruudessa [Aumasson, 2018]. Kaksiulotteinen tapaus nähdään kuvassa 10.



Kuva 10. 2-ulotteinen hila.

Hilapohjaiset kryptosysteemit perustuvat *lähimmän vektorin ongelmaan* (closest vector problem), missä ongelmana on löytää hilavektori, joka on lähimpänä annettua vektoriavaruuden pistettä (kuvassa 10 tähdenmuotoinen piste). Kun hilan kanta on tarpeeksi *ei-ortogonaalinen* eli kantavektoreiden *v* ja *w* välinen kulma on hyvin pieni, on tämän ongelman ratkaisu vaikea sekä klassisilla, että kvanttitietokoneilla.

NIST:n standardisointikilpailuun lähetetyistä 69 ehdotuksesta 24 pohjautuivat hilaongelmiin. Näistä ehdotuksista NTRU (Nth Degree Truncated Polynomial Ring Units) on kestänyt pisimmän ajan kryptanalyysiä. NTRU keksittiin vuonna 1996 kolmen matemaatikon Jeffrey Hoffsteinin, Joseph H. Silvermanin ja Jill Pipherin toimesta. NTRU:n kehittäjien kotisivulla mainitaankin, että kyseinen systeemi on 92 kertaa nopeampi salauksen purkamisessa kuin RSA samalla turvallisuuden tasolla ja 60% nopeampi salaamisessa kuin RSA. Samoin se on myös nopeampi kuin elliptisiin käyriin perustuvat salausmenetelmät ja salauksen purkumenetelmät. Täten se on vahva kandidaatti tulevaisuudessa uudeksi salausmenetelmien standardiksi. [NTRU Cryptosystem, 2019]

4.2. Koodipohjainen kryptografia

Koodipohjaiseen kryptografiaan kuuluvat sellaiset kryptosysteemit, jotka perustuvat *virheenkorjauskoodeihin* (error correcting codes). Kyseessä on informaatioteoreettinen ongelma, jossa tarkoituksena on lähettää tietoa virheellisen kommunikaatiokanavan läpi. Virheellisellä kanavalla tarkoitetaan sellaista kanavaa, jossa data saattaa vääristyä erilaisten häiriöiden toimesta. Jos esimerkiksi halutaan lähettää vastaanottajalle kolmen bitin verran tietoa, ”110” ja vastaanottaja saakin bitit ”111”, on tällöin tapahtunut yksi virhe. Yksinkertainen ratkaisu onkin monistaa jokainen bitti kolminkertaisesti, eli jos halutaan lähettää viesti ”110” tulisi se muuttaa viestiksi ”111111000”. Nyt viestin vastaanottaja tulkitsee jokaisen kolmen bitin jonon yhdeksi bitiksi, joka määrätään enemmistön mukaan. Tällöin viesti pystyy kestämään yhden bitin verran virhettä eli jos viesti muuttuisi esimerkiksi muotoon ”110111000”, tulkittaisiin se silti viestiksi ”110”. [Aumasson, 2018]

Lineaariset koodit ovat vähemmän triviaali tapa korjata virheitä. Lineaariset koodit tarkoittavat käytännössä vektoreiden kertomista matriiseilla ja tämä systeemi perustuu NP-täydellisenä tiedettyyn yleisen lineaarisen koodin dekoodauksen ongelmaan. Eräs tällainen kryptosysteemi on 1978-luvulla kehitetty *McEliecen kryptosysteemi*. [Aumasson, 2018]

4.3. Monimuuttujapohjainen kryptografia

Monimuuttujapohjaiset (multivariate based) kryptosysteemit perustuvat NP-vaikeaan ongelmaan nimeltä *monimuuttujaiset toisen asteen yhtälöt* (multivariate quadratics

equations). Otetaan esimerkiksi seuraava toisen asteen yhtälöryhmä, missä on neljä tuntematonta muuttujaa ja neljä yhtälöä:

$$x^2 + y + z = 3$$
$$xy + zw = 0$$
$$x^2 + y^2 - z - w = 5$$
$$z^2 + w = 2$$

Tämän yhtälöryhmän ratkaisuksi tulee siis löytää kaikille muuttujille *x*, *y*, *z* ja *w* arvot, joilla jokainen yhtälö pätee. [Aumasson, 2018]

Monimuuttujapohjaisia kryptosysteemeitä ei kuitenkaan tällä hetkellä käytetä suuremmissa sovelluksissa, sillä se on joko hidas tai paljon muistia vaativa systeemi. Lisäksi monet turvalliseksi ajatellut monimuuttujapohjaiset kryptosysteemit ovat osoittautuneet turvattomiksi, sillä turvallisten parametrien valinta on vaikeaa (kuinka monta yhtälöä, kuinka monta muuttujaa, numeroiden tyypit ja lukumäärät). [Aumasson, 2018]

4.4. Hajautuspohjainen kryptografia

Hajautuspohjaiset (hash-based) kryptosysteemit eroavat edellä mainituista siten, että ne perustuvat jo turvallisiksi todettuihin kryptografisiin hajautusfunktioihin. Hajautukseen perustuvia kryptografisia primitiivejä ei myöskään tällä hetkellä ole muita kuin digitaaliset allekirjoitukset, joiden avulla varmistetaan viestien aitous. Hajautuspohjaisia salausmenetelmiä ei vielä ole olemassa, mutta koska kvanttitietokoneet eivät pysty ”rikkomaan” kryptografisia hajautusfunktioita yhtään sen nopeammin kuin klassiset tietokoneet, hajautuspohjaiset digitaaliset allekirjoitukset ovat eräs kandidaatti kvanttitietokoneiden aikaisille allekirjoituksille. Hajautuspohjaisissa digitaalisissa allekirjoituksissa on myös sellainen rajoite, että ne pohjautuvat *Winternitzin yhden kerran allekirjoitukseen* (Winternitz one-time signature) eli niillä voi tuottaa vain yhden allekirjoituksen jokaista salaista avainta kohti. [Aumasson, 2018]

Esimerkkinä hajautuspohjaisesta digitaalisesta allekirjoitussysteemistä on SPHINCS, joka eroaa muista hajautuspohjaisista systeemeistä siten, että se on tilaton systeemi. Tällä tarkoitetaan sitä, että muut hajautuspohjaiset allekirjoitussysteemit päivittävät salaista avainta, jolla allekirjoitus muodostetaan. SPHINCS taas pystyy käyttämään samaa avainta eri allekirjoitusten välillä. SPHINCS myös vähentää digitaalisten allekirjoitusten kokoa pienentämällä sisäisen puutietorakenteensa kokoa ilman että turvallisuustasosta täytyy karsia. [Bernstein et al. 2017]

5. Käytännön vaikutukset

Vaikka kvanttitietokoneet eivät suurella skaalalla olekaan vielä tätä päivää, tärkeää on jo nyt pohtia kvanttitietokoneiden vaikutusta käytännössä. Kuten arvostettu kryptograafikko Daniel J. Bernstein [2009] totesi, ”jos odotamme siihen asti, kunnes kvanttitietokoneet ovat suurella skaalalla arkipäivää, on vuosien tutkimustyö mennyt hukkaan”. Tutkimustyötä tarvitaan, sillä kvanttiresistantteihin kryptosysteemeihin käytetty tutkimustyö on paljon pienempi kuin esimerkiksi RSA:han tai Diffie-Hellmaniin käytetty työ. Ajan myötä kvanttiresistantit kryptosysteemit paranevat sekä tehokkuudeltaan, että turvallisuudeltaan, sillä nykyään ne eivät ole vielä valmiita suuren skaalan tuotantokäyttöön.

Tässä luvussa käydään läpi ohjelmistotalo Solitan keskeisimpien ohjelmistojen ja laitteistojen kryptosysteemien tila, sekä pohditaan, miten kyseisten systeemien tulisi varautua suuren skaalan kvanttitietokoneiden aikaan.

5.1. Microsoft Outlook Office 365

Sähköpostien lukemiseen ja lähettämiseen, sekä kalenterinhallintaan käytetty Microsoftin Outlook Office 365 -sovellus salaa sähköpostipalvelinten välisen kommunikaation TLS 1.2 -protokollalla (Transport Layer Security) [Microsoft Outlook Office 365, 2019]. TLS-protokolla, entinen SSL, tunnetaan ehkä parhaiten HTTPS-protokollan mukana olemisesta. Tällä protokollalla suojataan suurin osa internetin liikenteestä tänä päivänä. TLS käyttää avainten vaihtoon elliptisten käyrien Diffie-Hellman -protokollaa, joka on myös ratkottavissa Shorin algoritmilla ja näin ollen on turvaton kvanttitietokoneiden aikakaudella.

5.2. Slack

Slack on Slack Technologies:n kehittämä, Solitan sisäisesti käyttämä pikaviestintäsovellus, jonka avulla työntekijät voivat muun muassa lähettää keskenään pikaviestejä, soittaa videopuheluita sekä jakaa tiedostoja. Slack:n turvallisuusjulkaisun (Security White Paper) mukaan kaikki liikenne asiakassovelluksen (client) ja palvelimen välillä on salattu TLS 1.2 -protokollalla.

5.3. SSH

SSH (Secure Shell) on komentorivityökalu, jonka avulla voidaan kirjautua etänä toiselle tietokoneelle salatun yhteyden avulla. SSH:ta käytetään lähes jokaisessa projektissa muun muassa palvelimen konfiguroinnin yhteydessä, palvelimen lokien lukemisen yhteydessä, sovellusten uudelleenkäynnistyksien yhteydessä jne. SSH käyttää julkisen avaimen kryptosysteemejä salatun yhteyden muodostamiseen. SSH tarjoaa vaihtoehdoiksi sekä

RSA:ta, klassista Diffie-Hellmania että elliptisten käyrien Diffie-Hellmania avainten vaihtoon.

5.4. Git

Git on hajautettu versionhallintatyökalu, jonka avulla pidetään huoli koodipohjan eheänä pysymisestä usean kehittäjän työskennellessä samaan aikaan sen parissa. Git:ä käytetään myös lähes jokaisessa projektissa Solitalla. Koska Gitin kautta kulkee sovelluskoodi, äärimmäisen tärkeää on, että yhteys on vahvasti salattu. Git tarjoaa neljää protokollaa tiedonsiirtoon: HTTP, HTTPS, SSH ja Gitin oma Git-protokolla. Näistä vain HTTPS ja SSH ovat salattuja, sillä avoimen lähdekoodin projektien osalta tiedonsiirron salauksesta ei olisi paljon hyötyä. [Git protocols, 2019]

5.5. Fortigate VPN Client

Fortigate VPN Client on VPN (Virtual Private Network) -sovellus, jonka avulla Solitan työntekijät pystyvät myös toimistolta poissa ollessaan pääsemään käsiksi firman sisäverkossa oleviin resursseihin. Fortigate tukee vain symmetrisiä kryptosysteemejä, kuten AES ja DES, jolloin työntekijän koneen ja VPN-palvelimen on jaettava yksityiset avaimet etukäteen. Työntekijän koneella onkin ennalta asennettu sertifikaatti, jonka avulla salatun kommunikaatiokanavan muodostaminen tapahtuu. [Fortigate IPsec encryption, 2018]

5.6. Cisco ASA

Solitalla on myös mahdollista muodostaa ulkopuolisten toimijoiden verkkojen välillä L2L (Lan-to-Lan) yhteyksiä. Tähän käytettävä Cisco ASA-laite (Adaptive Security Appliances) voi muodostaa muun muassa IPSec-yhteyden (IP Security Architecture) osapuolten välille. IPSec-protokollan yhteydessä taasen käytetään IKE-avaintenvaihtoprotokollaa (Internet Key Exchange), joka tunnelinmuodostuksen yhteydessä määrittää muun muassa käytettävän Diffie-Hellman ryhmän, jota käytetään avaimenvaihdossa. Diffie-Hellman ryhmä määrittää avaintenvaihdossa luotavan avaimen pituuden. [Cisco ASA, 2018]

5.7. Suositeltavat toimenpiteet

Kvanttikoneiden aikaan varautuakseen on hyvä tiedostaa, mitkä kryptosysteemit ovat alttiita kvanttitietokoneille sekä mitkä systeemit tarjoavat nykytiedon valossa turvallisen vaihtoehdon nykysysteemeille. Kolmannen osapuolen sovelluksia ja palveluita käyttäessä ei itse voi vaikuttaa käytettäviin systeemeihin muuten kuin odottamalla, että kyseiset palveluiden toimittajat integroivat uudet turvalliset systeemit palveluihinsa ja tämän jälkeen ottamalla nämä käyttöön. Tässä alaluvussa pohditaan, miten edellä

mainitut sovellukset/laitteistot voisivat tulevaisuudessa päivittää käyttämänsä kryptosysteemit kvanttitietokoneaikaa ajatellen turvallisempiin systeemeihin.

Outlook, Slack, sekä Git käyttävät jokainen ”konepellin alla” TLS-protokollaa datan suojaamiseen. Outlook ja Slack käyttävät tätä suoraan, kun taas Git käyttää tätä HTTPS-protokollan välityksellä. TLS taas tarjoaa avaintenvaihtoprotokollaksi liudan eri vaihtoehtoja, joihin kuuluu muun muassa RSA, DH ja ECDH (Elliptic Curve Diffie Hellman). Nämä ovat kuitenkin kaikki alttiita Shorin algoritmille, joten TLS:n tulisi tarjota tulevaisuuden versioissaan kvanttiresistantteja vaihtoehtoja avaimenvaihdolle. Lupaavimmat vaihtoehdot kirjoitushetkellä ovat muun muassa NewHope [NewHope, 2018], joka on hilaongelmiin pohjautuva avaimenvaihtoprotokolla hyvällä suorituskyvyllä ja turvallisuustasolla, NTRU, joka myös perustuu hilaongelmiin tai SIDH (Supersingular isogeny Diffie–Hellman key exchange), joka liittyy vahvasti elliptisiin käyriin. Näitä on vertailtu muun muassa Vladimir Valyukhin [2017] tutkielmassa “Performance and comparison of post-quantum cryptographic algorithms”.

Git voi käyttää myös SSH:ta, joka puolestaan tarjoaa myös vaihdoehdoiksi RSA:n, DH:n ja ECDH:n. Samat suositukset pätevät siis tässäkin kuin yllä. Cisco ASA -laite käytti IPSec-yhteyden muodostukseen IKE-protokollaa, joka käyttää Diffie-Hellmania myös, joten tässäkin pätee systeemin korvaaminen jollakin edellä mainituista kolmesta kvanttiresistantista kryptosysteemistä.

Fortigate VPN -client käyttää symmetristä kryptosysteemiä asymmetrisen sijaan eli salausavain on jaettu osapuolten välillä etukäteen. Fortigate tarjoaa joko AES- tai DES-kryptosysteemeitä datan salaamiseen. Groverin algoritmin avulla kvanttitietokoneet vähentävät salausavainten turvallisuutta neliöjuureen, eli esimerkiksi 256 bitin turvallisuustaso klassisia tietokoneita vastaan vähenee 128 bittiin kvanttitietokoneita vastaan. Tähän on onneksi olemassa yksinkertainen ratkaisu: tuplataan salausavainten koko 512 bittiin. Näin turvallisuustaso pysyy edelleen 256 bitissä. Tämä ohje pätee toki myös julkisen avaimen kryptosysteemeihin, sillä Groverin algoritmin avulla voidaan myös etsiä julkisen avaimen kryptosysteemeiden avulla luotu jaettu salainen avain.

Konkreettisia implementaatioita kvanttiresistanteille kryptosysteemeille on kirjoitushetkellä muun muassa Open Quantum Safe -projektin toimesta tehty liboqs-kirjasto. Kyseessä on C-kielellä kirjoitettu avoimen lähdekoodin kirjasto, johon on implementoitu NIST:n standardisointikilpailun tuottamia kvanttiresistantteja kryptosysteemeitä, muun muassa NewHope ja SIDH. Kirjastoa on myös mahdollista käyttää C#, C++ ja Python -ohjelmointikielillä, sekä tuki Javalle on tulossa. Kirjoitushetkellä kirjasto on integroitu muun muassa OpenSSL:n, sekä OpenSSH:n kanssa omissa versiohaaroissaan, jolla tarjotaan kvanttiresistantit vaihtoehdot TLS ja SSH -protokollille. Kirjastoa on myös käytetty Microsoftin kvanttiresistantin VPN-

projektin yhteydessä, missä tarkoituksena on selvittää kvanttiresistanttien kryptosysteemien toimivuus VPN-ympäristössä sekä Mullvad-nimisen yrityksen VPN:sä. Kirjasto vaikuttaa siis hyvin lupaavalta korvaamaan kaikki Solitalla käytössä olevien sovellusten käyttämät, kvanttikoneille alttiina olevat salausprotokollat. Outlook, Slack ja Git käyttävät kaikki TLS- tai SSH-protokollaa salaukseen. Fortigate VPN:n voisi korvata tulevaisuudessa esimerkiksi Microsoftin tarjoamalla kvanttiresistantilla VPN:llä. [Open Quantum Safe, 2019]

Täytyy kuitenkin muistaa, että satunnaisella hyökkääjällä on muitakin esteitä kuin pelkkä salausavainten murtaminen. Ensinnäkin hänen tulisi päästä liittymään sisäverkkoon tai jotenkin muuten kierrättämään yhteys oman tietokoneensa kautta, jotta hän pääsisi lukemaan ja tallentamaan liikennettä. Tämän saavuttaakseen hänen tulisi päästä palomuurista läpi ja kiertää kaikki muu sisäverkon suojaus. Jos kuitenkin oletetaan tämän olevan mahdollista, hyökkääjä voi liikenteen lukemisen ja tallentamisen avulla varastaa nyt dataa, jonka salausta ei vielä nykyään pystytä purkamaan ja myöhemmin kvanttitietokoneen avulla murtaa salauksen. Salausalgoritmien vaihtaminen jossain myöhemmässä vaiheessa ei enää auta aiemmin varastetun datan salaamiseen, vaan tällaisen datan osalta toivo on jo menetetty. Yksi tapa suojata kriittistä dataa tätä vastaan on datan pitäminen verkon ulkopuolella, mitä harrastetaankin Solitalla. Tässä tilanteessa kvanttitietokoneen omistaminen ei auta mitään, kun dataan ei yksinkertaisesti pääse käsiksi. Tämä yksinkertaiselta kuulostava suojautumiskeino saattaakin lopulta olla paras keino kvanttitietokoneilta suojautumista varten, varsinkin kun kyseessä on erittäin kriittinen, esimerkiksi julkishallinnon turvaluokiteltu data.

6. Yhteenveto

Tässä tutkielmassa määriteltiin kryptografia ja esiteltiin modernin kryptografian tärkeimpiä ja käytetyimpiä rakennuspalikoita, kuten julkisen avaimen kryptosysteemit RSA ja Diffie-Hellman. Tutkielmassa määriteltiin kvanttitietokone, vertailtiin sitä klassiseen tietokoneeseen ja esiteltiin tunnetuimmat kvanttialgoritmit Deutsch-Jozsa, Grover ja Shor sekä kerrottiin, miten nämä uhkaavat modernia kryptografiaa. Tämän uhan takia tutkijoiden mielenkiinto on kääntynyt kvanttiresistanttien kryptosysteemien suuntaan ja muun muassa NIST:n julkaisemasta standardisointikilpailusta kerrottiin. Lopulta selvitettiin ohjelmistotalo Solitan käyttämien, modernia kryptografiaa hyödyntävien sovellusten käyttämiä kryptosysteemeitä ja etsittiin mahdollisia korvaajia tulevaisuuden kvanttitietokoneiden uhkaa silmällä pitäen. Eräs lupaava projekti nimeltä OpenQuantumSafe löytyi ja sen puitteissa kirjoitettu kvanttiresistantteja kryptosysteemejä hyödyntävä kirjasto liboqs on mahdollinen tulevaisuuden korvaaja monille sovelluksille. Kvanttitietokoneet eivät kuitenkaan vielä vähään aikaan ole siinä pisteessä, että kryptosysteemien korvaaminen olisi ajankohtaista. On kuitenkin tärkeää jo nyt tutkia erilaisia vaihtoehtoja, sillä kryptosysteemit vaativat tunnetusti monen vuoden ajalta kryptanalyysiä, jotta niihin voisi luottaa tuotantokäytössä.

Aiheen ollessa varsin laaja, on jatkotutkimuskohteitakin varsin monta. Yksi mielenkiintoinen kysymys liittyy kvanttialgoritmeihin ja niiden tilavaatimuksen pienentämiseen ajoajan kustannuksella, jolloin vaadittaisiin vähempi määrä kubitteja samojen ongelmien ratkaisemiseen. Kuinka pieneksi olisi mahdollista saada Shorin algoritmin vaatima kubittien määrä, jotta esimerkiksi 2048-bittinen RSA-avain saataisiin murrettua? Toinen tutkimuskohde voisi olla universaalien, kvanttiportteihin perustuvien kvanttitietokoneiden, ja kvanttihehkutukseen perustuvien kvanttitietokoneiden välinen vertailu. Millaisia ongelmia molemmat ratkovat paremmin kuin toinen ja kuinka haastavaa on molempien rakentaminen? Kolmas tutkimuskohde voisi olla kvanttiresistanttien kryptosysteemien vertailu keskenään niin suorituskyvyn, turvallisuuden kuin kompleksisuuden osalta ja esimerkiksi liboqs-kirjaston arviointi käytännössä.

Viiteluettelo

Aumasson, Jean-Philippe. 2018. *Serious Cryptography: A Practical Introduction to Modern Encryption.* No Starch Press.

Bernstein, Daniel J. et al. 2009. *Post-Quantum Cryptography*. Springer-Verlag.

Bernstein, Daniel J. et al. 2015. *SPHINCS: practical stateless hash-based signatures*. Dept. of Computer Science, University of Illinois at Chicago.

Bernstein, Daniel J. et al. 2017. *A low-resource quantum factoring algorithm*. Dept. of Computer Science, University of Illinois at Chicago.

Boneh, Dan. 1999. *Twenty Years of Attacks on the RSA Cryptosystem*. Notices of the American Mathematical Society (AMS). 46 (2): 203-213.

Cisco ASA. 2018. *https://www.cisco.com/c/en/us/td/docs/security/asa/asa90/configuration/guide/asa_90_cli_config/vpn_ike.html* (viitattu 23.3.2019)

Daemen, Joan & Rijmen, Vincent. 1998. *The Block Cipher Rijndael*. Lecture Notes in Computer Science - LNCS. 1820. 277-284.

Damgård, Ivan B. 1989. *A Design Principle for Hash Functions*. Advances in cryptology - CRYPTO ’89. Proceedings of a conference held at the University of California, Santa Barbara, CA (USA).

Davies-Meyer construction. 2019. *https://www.cs.rit.edu/~ark/462/module11/fig1.png* (viitattu 20.5.2019)

Deutsch, D & Jozsa, R. 1992. *Rapid Solution of Problems by Quantum Computation*. Proc. Roy. Soc. Lond. A. 439. 553-558.

Diffie, W & Hellman, M. 1976. *New directions in cryptography*. IEEE Transactions on Information Theory. 22 (6): 644–654.

D-Wave 2000Q. 2017. *https://www.dwavesys.com/press-releases/d-wave%c2%a0announces%c2%a0d-wave-2000q-quantum-computer-and-first-system-order* (viitattu 16.2.2019)

D-Wave Quantum Annealing. 2016. *https://www.theverge.com/2016/9/28/13057414/quantum-computer-d-wave-2000-qubit-chip* (viitattu 16.2.2019)

Fortigate IPsec encryption. 2018. *https://help.fortinet.com/fos50hlp/54/Content/FortiOS/fortigate-ipsecvpn-54/IPsec_VPN_Concepts/Encryption.htm* (viitattu 17.3.2019)

Git protocols. 2019. *https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols* (viitattu 17.3.2019)

Google Bristlecone. 2018. *https://www.sciencenews.org/article/google-moves-toward-quantum-supremacy-72-qubit-computer* (viitattu 16.2.2019)

Grover, Lov K. 1996. *A fast quantum mechanical algorithm for database search*. Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC).

IBM quantum computer. 2019. *https://www.newscientist.com/article/2189909-ibm-unveils-its-first-commercial-quantum-computer/* (viitattu 16.2.2019)

Intel Tangle Lake. 2018. *https://spectrum.ieee.org/tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy* (viitattu 16.2.2019)

Martín-López, Enrique & Laing, Anthony & Lawson, Thomas & Alvarez, Roberto & Zhou, Xiaoqi & O'Brien, Jeremy. 2012. *Experimental realisation of Shor's quantum factoring algorithm using qubit recycling*. Centre for Quantum Photonics, H. H. Wills Physics Laboratory & Department of Electrical and Electronic Engineering, University of Bristol.

Merkle-Damgård construction. 2019. *https://www.researchgate.net/profile/Yevgeniy_Dodis/publication/221355367/figure/fig3/AS:650858744266761@1532188225346/The-plain-Merkle-Damgard-Construction.png* (viitattu 20.5.2019)

Microsoft Outlook Office 365. 2019. *https://docs.microsoft.com/en-us/office365/securitycompliance/exchange-online-uses-tls-to-secure-email-connections* (viitattu 17.3.2019)

Monz, Thomas & Schindler, Philipp & T Barreiro, Julio & Chwalla, Michael & Nigg, Daniel & Coish, W & Harlander, Maximilian & Hänsel, Wolfgang & Hennrich, Markus & Blatt, R. 2010. *14-qubit entanglement: creation and coherence*. Institut für Experimentalphysik, Universität Innsbruck.

NewHope. 2018. *https://newhopecrypto.org/* (viitattu 29.3.2019)

Nielsen, Michael A. & Chuang, Isaac L. 2010. *Quantum Computation and Quantum Information, 10th anniversary edition.* Cambridge University Press.

NTRU Cryptosystem. 2019. *https://www.onboardsecurity.com/products/ntru-crypto* (viitattu 17.2.2019)

Open Quantum Safe. 2019. *https://openquantumsafe.org* (viitattu 6.4.2019)

Preskill, John. 2011. *Quantum Computing and the Entanglement Frontier*. Rapporteur talk at the 25th Solvay Conference on Physics "The Theory of the Quantum World" Brussels, 19-22 October 2011.

Rivest, Ronald & Shamir, Adi & M. Adleman, Leonard. 1978. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. Communications of the ACM. 21 (2): 120–126.

Shor, Peter W. 1996. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. IEEE Computer Society Press.

Valyukh, Vladimir. 2017. *Performance and comparison of post-quantum cryptographic algorithms*. Master's thesis. Linköping University, Department of Electrical Engineering.

Van Assche, Gilles. 2006. *Quantum Cryptography and Secret-Key Distillation*. Cambridge University Press.

Wang, Xi-Lin et al. 2018. *18-qubit entanglement with photon's three degrees of freedom*. Hefei National Laboratory for Physical Sciences at Microscale and Department of Modern Physics, University of Science and Technology of China, & CAS-Alibaba Quantum Computing Laboratory, CAS Centre for Excellence in Quantum Information and Quantum Physics, University of Science and Technology of China, Shanghai 201315, China.

Washington, Lawrence C. 2008. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. CRC Press.

Wehner, Sam. 2017. *Quantum Computing: An Intuitive Explanation of Grover's Algorithm. https://www.linkedin.com/pulse/quantum-computing-intuitive-explanation-grovers-algorithm-sam-wehner* (viitattu 26.4.2019)